

AFIT/DS/ENG/98-10

**New Algorithms for Moving-Bank Multiple Model
Adaptive Estimation**

DISSERTATION

**Juan R. Vasquez
Captain, USAF**

AFIT/DS/ENG/98-10

19980629 029

Approved for public release; distribution unlimited

Disclaimer

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

AFIT/DS/ENG/98-10

New Algorithms for Moving-Bank Multiple Model Adaptive Estimation

DISSERTATION

Presented to the Faculty of the Graduate School of Engineering of the Air Force Institute of
Technology Air University In Partial Fulfillment for the Degree of

Doctor of Philosophy

Juan R. Vasquez
Captain, USAF

Air Force Institute of Technology

Wright-Patterson AFB, Ohio

May 26, 1998

Approved for public release; distribution unlimited

New Algorithms for Moving-Bank Multiple Model Adaptive Estimation

Juan R. Vasquez

Captain, USAF

Approved:

Peter S. Maybeck

Dr. Peter S. Maybeck
Committee Chairman

26 May 98

Date

M. Pachter

Dr. Meir Pachter
Committee member

26 May 98

Date

Mark E. Oxley

Dr. Mark E. Oxley
Committee member

26 May 98

Date

William E. Wiesel, Jr.

Dr. William E. Wiesel, Jr.
Dean's Representative

26 May 98

Date

Accepted:

Robert A. Calico, Jr.

Dr. Robert A. Calico, Jr.
Dean, School of Engineering

Acknowledgments

I would like to thank two members of my committee, Dr. Mark Oxley and Dr. Meir Pachter, for their continued support and constructive insights. They not only possess superior intellect, but are blessed with the ability and willingness to pass on their knowledge to their students. I would also like to recognize Don Smith for keeping my computer resources alive and well. These pages would be blank without his help. My endless gratitude and respect are given to my advisor, Dr. Peter Maybeck. His exceptional intelligence is only exceeded by his personal integrity and regard for his students. He is without a doubt the finest instructor (both in and out of the classroom) that I have had the honor of knowing, and his continued encouragement was fundamental to my success. I am proud to say that I was his student. Finally, I would like to let my wife, Angela, know how much her love and understanding has meant to me. I will always remember her driving me to those dreadful exams (the BIG exams) while filling me with encouragement. This dissertation is a wonderful accomplishment for me, but it would be meaningless without her here to share in my joy.

Juan R. Vasquez

Table Of Contents

	Page
Acknowledgments	iv
Table Of Contents	v
List Of Figures	x
List Of Tables	xvii
Abstract	xix
Chapter 1. INTRODUCTION	1
1.1 Overview	1
1.2 Dissertation Overview	10
Chapter 2. BACKGROUND CONCEPTS	11
2.1 Parameter Estimation and Failure Detection	12
2.1.1 Voting Method	12
2.1.2 Chi-Square Test and the Kalman Filter	13
2.1.3 Sequential Probability Ratio Test (SPRT)	16
2.1.4 Generalized Likelihood Ratio (GLR) Test	18
2.1.5 Distributed Kalman Filtering (DKF)	24
2.1.6 Multiple Model Adaptive Estimation (MMAE)	25
2.2 MMAE Review	27
2.2.1 MMAE Fundamentals	28

2.2.2	Simple Performance Enhancements	32
2.2.3	Moving-Bank MMAE	38
2.2.3.1	Decision Logics	39
2.2.3.2	Sheldon Discretization	43
2.2.4	Hierarchical Structure	50
2.2.5	Inter-Residual Distance Feedback (IRDF)	52
2.2.6	Model-Group Switching	53
2.3	Chapter Summary	54
Chapter 3.	THEORY DEVELOPMENT	55
3.1	Density Algorithm	56
3.1.1	Algorithm Development	57
3.1.2	Candidate Measures Rejected	60
3.1.3	Candidate Measures Accepted	66
3.1.4	Algorithm Description	79
3.1.4.1	Main Routine	80
3.1.4.2	Move Subroutine	95
3.1.5	Performance Enhancements	102
3.1.5.1	Dead Zone for Soft Moves	103
3.1.5.2	Initialization of Newly Declared Filters	105
3.1.5.3	Decision Delays	106

3.1.5.4	On-Line Sheldon Discretization	107
3.2	Probability Algorithm	109
3.2.1	Probability Calculation and Coordinate Transformation	112
3.2.2	Multi-Dimensional Numerical Integration	120
3.2.3	Discretization Method	122
3.2.3.1	Parameter Value Search Routine	126
3.2.3.2	Implementation Issues	132
3.2.4	Algorithm Description	134
3.2.5	Dead Zone for Moves	134
3.3	Combined Density and PBDM Algorithm	135
3.4	Chapter Summary	136
Chapter 4.	SIMULATION PERFORMANCE	137
4.1	Overview	137
4.2	System Description	138
4.2.1	INS Models	139
4.2.1.1	The INS Truth and Filter Models	139
4.2.1.2	The INS Measurement Model	141
4.2.2	The Radar Altimeter Model	141
4.2.3	GPS Models	142
4.2.3.1	The 30-State GPS System Model	142

4.2.3.2	The GPS Truth Model and Filter Design Models	145
4.2.3.3	The GPS Measurement Model	145
4.3	Event Models	148
4.4	Data Presentation	152
4.5	Simulation Software	154
4.6	Fixed Bank MMAE	155
4.6.1	Implementation Issues	155
4.6.2	Performance	156
4.7	MMAE Incorporating the Density Algorithm	178
4.7.1	Implementation Issues	178
4.7.2	Performance <i>Without</i> Expansion and Additional Delay	186
4.7.3	Performance <i>With</i> Expansion and Additional Delay	201
4.8	MMAE Incorporating the Density Algorithm with Sheldon Discretization	211
4.8.1	Implementation Issues	211
4.8.2	Performance <i>Without</i> Expansion and Additional Delay	216
4.8.3	Performance <i>With</i> Expansion and Additional Delay	225
4.9	MMAE Incorporating the Probability Algorithm	239
4.9.1	Implementation Issues	239
4.9.2	Performance	241
4.10	MMAE Incorporating the Density Algorithm with Probability Discretization	249

4.10.1	Implementation Issues	249
4.10.2	Performance	253
4.11	Baseline Without Interference/Jamming	259
4.12	Single Kalman Filter	268
4.13	Chapter Summary	268
Chapter 5.	CONCLUSIONS AND RECOMMENDATIONS	271
5.1	Conclusions	271
5.2	Recommendations	276
Appendix A.	DERIVATION OF DENSITY MOMENTS	280
Appendix B.	NUMERICAL INTEGRATION	282
Appendix C.	EQUATIONS FOR THE MEAN OF THE RESIDUAL	287
C.1	Mismodeled Input Matrix	288
C.2	Mismodeled Output Matrix	289
C.3	Mismodeled State Transition Matrix	289
C.4	Mismodeling Measurement or Dynamics Noise Covariances	289
Appendix D.	MODEL STATE DEFINITIONS AND SYSTEM MATRICES	292
Appendix E.	TABULATED PERFORMANCE MEASURES	295
Bibliography	298
Vita	304

List Of Figures

	Page
Figure 1. Generic State and Parameter Estimator	2
Figure 2. Multiple Model Adaptive Estimation Algorithm	3
Figure 3. Moving-Bank MMAE for a Two-Dimensional Parameter Space	5
Figure 4. <i>Moved</i> Moving-Bank MMAE	6
Figure 5. <i>Expanded</i> Moving-Bank MMAE	7
Figure 6. Conventional MMAE	8
Figure 7. M ³ AE Architecture	9
Figure 8. Multi-Sensor Navigation System	12
Figure 9. Multiple GLR Testing	23
Figure 10. Multiple Model Adaptive Estimation Algorithm	26
Figure 11. Hierarchical Modeling - Level 0 and Level 1 MMAE Banks	51
Figure 12. Moving-Bank MMAE Topics of Interest	56
Figure 13. All Density Values Relatively Large	60
Figure 14. All Density Values Relatively Small	61
Figure 15. Large Variations in Density Values	63
Figure 16. Spline Fit of Density Samples	64
Figure 17. Illustration of Scale Factor in Calculating Measure M_6	71
Figure 18. Finely Discretized Filters \Rightarrow Expand the Bank	72

Figure 19.	Coarsely Discretized Filters \Rightarrow Contract the Bank	73
Figure 20.	Chi-Squared Density for $m=6$ Degrees of Freedom	76
Figure 21.	Density Algorithm Main Routine	81
Figure 22.	Move MMAE Bank to the Left	82
Figure 23.	Redistribute Filter Bank	84
Figure 24.	Contraction of Filter Bank	86
Figure 25.	Filter Bank After Contraction	87
Figure 26.	Filter Bank Prior to Contraction	87
Figure 27.	Alternative Spacing of New Parameter Values for Contractions	90
Figure 28.	Uniform Versus Actual Filter-Assumed Parameter Values with Interpolation	92
Figure 29.	Expansion of Filter Bank	93
Figure 30.	Density Algorithm Move Subroutine	96
Figure 31.	Types of Bank Movements (Soft, Medium, and Hard)	98
Figure 32.	Density Function Contour Map (Hyper)ellipsoids for Two- Dimensional Residuals	115
Figure 33.	<i>Transformed</i> Density Function Contour Map (Hyper)spheroids for Two-Dimensional Residuals	116
Figure 34.	<i>Transformed</i> Density Function Contour Map (Hyper)spheroids for Two-Dimensional Residuals with a Non-Zero Mean	117
Figure 35.	Example Integration Curves with Equal Probability Variation	125
Figure 36.	Parameter Value Search Routine	128

Figure 37.	PLS Block Diagram	139
Figure 38.	Simulated Flight Profile	151
Figure 39.	Autocorrelation Curve for Constrained-Range Parameter Discretization	157
Figure 40.	Parameter Estimation Performance – Case 1: Fixed-Bank	158
Figure 41.	MMAE State Estimation Errors (feet) – Case 1: Fixed-Bank	160
Figure 42.	Elemental Filter Probabilities – Case 1: Fixed-Bank	162
Figure 43.	Elemental Filter Probabilities Mean ± 1 Sigma – Case 1: Fixed-Bank	163
Figure 44.	M ³ AE State Estimation Errors (feet) – Case 1: Fixed-Bank	164
Figure 45.	Parameter Estimate ($3700 \text{ sec} \leq t < 3750 \text{ sec}$) – Case 1: Fixed-Bank	167
Figure 46.	Parameter Estimation Performance – Case 2: Fixed-Bank	170
Figure 47.	Elemental Filter Probabilities – Case 2: Fixed-Bank	171
Figure 48.	Parameter Estimation Performance – Case 3: Fixed-Bank	172
Figure 49.	Parameter Estimation Performance – Case 4: Fixed-Bank	174
Figure 50.	Elemental Filter Probabilities – Case 4: Fixed-Bank	175
Figure 51.	MMAE State Estimation Errors (feet) – Case 4: Fixed-Bank	176
Figure 52.	M ³ AE State Estimation Errors (feet) – Case 4: Fixed-Bank	177
Figure 53.	Parameter Estimation Performance – Case 5: Fixed-Bank	179
Figure 54.	Elemental Filter Probabilities – Case 5: Fixed-Bank	180
Figure 55.	MMAE State Estimation Errors (feet) – Case 5: Fixed-Bank	181
Figure 56.	M ³ AE State Estimation Errors (feet) – Case 5: Fixed-Bank	182

Figure 57.	Parameter Estimation Performance – Case 1: Density Algorithm	187
Figure 58.	Elemental Filter Probabilities – Case 1: Density Algorithm	188
Figure 59.	MMAE State Estimation Errors (feet) – Case 1: Density Algorithm	190
Figure 60.	M ³ AE State Estimation Errors (feet) – Case 1: Density Algorithm	191
Figure 61.	Parameter Estimation Performance – Case 2: Density Algorithm	192
Figure 62.	Elemental Filter Probabilities – Case 2: Density Algorithm	193
Figure 63.	Parameter Estimation Performance – Case 4: Density Algorithm	195
Figure 64.	Elemental Filter Probabilities – Case 4: Density Algorithm	196
Figure 65.	MMAE State Estimation Errors (feet) – Case 4: Density Algorithm	199
Figure 66.	M ³ AE State Estimation Errors (feet) – Case 4: Density Algorithm	200
Figure 67.	Parameter Estimation Performance – Case 5: Density Algorithm	202
Figure 68.	Elemental Filter Probabilities – Case 5: Density Algorithm	203
Figure 69.	MMAE State Estimation Errors (feet) – Case 5: Density Algorithm	204
Figure 70.	M ³ AE State Estimation Errors (feet) – Case 5: Density Algorithm	205
Figure 71.	Parameter Estimation Performance – Case 1: Density Alg. with Expansions / Delay	206
Figure 72.	Elemental Filter Probabilities – Case 1: Density Alg. with Expansions / Delay	208
Figure 73.	M ³ AE State Estimation Errors (feet) – Case 1: Density Alg. with Expansions / Delay	209
Figure 74.	Parameter Estimation Performance – Case 4: Density Alg. with Expansions / Delay	210

Figure 75.	MMAE State Estimation Errors (feet) – Case 4: Density Alg. with Expansions / Delay	212
Figure 76.	M ³ AE State Estimation Errors (feet) – Case 4: Density Alg. with Expansions / Delay	213
Figure 77.	Parameter Estimation Performance – Case 5: Density Alg. with Expansions / Delay	214
Figure 78.	Parameter Estimation Performance – Case 1: Density / Sheldon Alg.	217
Figure 79.	Parameter Estimation Performance – Case 2: Density / Sheldon Alg.	219
Figure 80.	Parameter Estimation Performance – Case 3: Density / Sheldon Alg.	220
Figure 81.	Elemental Filter Probabilities – Case 3: Density / Sheldon Alg.	221
Figure 82.	Parameter Estimation Performance – Case 4: Density / Sheldon Alg.	223
Figure 83.	Parameter Estimation Performance – Case 5: Density / Sheldon Alg.	224
Figure 84.	Elemental Filter Probabilities – Case 5: Density / Sheldon Alg.	226
Figure 85.	MMAE State Estimation Errors (feet) – Case 5: Density / Sheldon Alg.	227
Figure 86.	M ³ AE State Estimation Errors (feet) – Case 5: Density / Sheldon Alg.	228
Figure 87.	Parameter Estimation Performance – Case 1: Density / Sheldon / Expansions / Delay	229
Figure 88.	Elemental Filter Probabilities – Case 1: Density / Sheldon Alg. / Expansions / Delay	230
Figure 89.	Parameter Estimation Performance – Case 7: Density / Sheldon / Expansions / Delay	233
Figure 90.	Parameter Estimation Performance – Case 2: Density / Sheldon / Expansions / Delay	234
Figure 91.	Elemental Filter Probabilities – Case 2: Density / Sheldon / Expansions / Delay	235

Figure 92.	Parameter Estimation Performance – Case 5: Density / Sheldon / Expansions / Delay	237
Figure 93.	Elemental Filter Probabilities – Case 5: Density / Sheldon / Expansions / Delay	238
Figure 94.	Parameter Estimation Performance – Case 1: Probability Algorithm	242
Figure 95.	MMAE State Estimation Errors (feet) – Case 1: Probability Algorithm	243
Figure 96.	M ³ AE State Estimation Errors (feet) – Case 1: Probability Algorithm	245
Figure 97.	Parameter Estimation Performance – Case 2: Probability Algorithm	246
Figure 98.	Parameter Estimation Performance – Case 4: Probability Algorithm	248
Figure 99.	MMAE State Estimation Errors (feet) – Case 4: Probability Algorithm	250
Figure 100.	M ³ AE State Estimation Errors (feet) – Case 4: Probability Algorithm	251
Figure 101.	Parameter Estimation Performance – Case 5: Probability Algorithm	252
Figure 102.	Parameter Estimation Performance – Case 1: Density / PBDM	254
Figure 103.	Elemental Filter Probabilities – Case 1: Density / PBDM	255
Figure 104.	MMAE State Estimation Errors (feet) – Case 1: Density / PBDM	256
Figure 105.	M ³ AE State Estimation Errors (feet) – Case 1: Density / PBDM	257
Figure 106.	Parameter Estimation Performance – Case 2: Density / PBDM	258
Figure 107.	Parameter Estimation Performance – Case 6: No Jamming	260
Figure 108.	Elemental Filter Probabilities – Case 6: No Jamming	261
Figure 109.	MMAE State Estimation Errors (feet) – Case 6: No Jamming	262
Figure 110.	M ³ AE State Estimation Errors (feet) – Case 6: No Jamming	263

Figure 111.	MMAE State Estimation Errors (feet) – Case 6: No Jamming and Modified Blending	265
Figure 112.	M ³ AE State Estimation Errors (feet) – Case 6: No Jamming and Modified Blending	266
Figure 113.	Elemental Filter Probabilities – Case 6: No Jamming and Modified Blending	267
Figure 114.	State Estimation Errors (feet) – Case 1: Single Kalman Filter	269
Figure 115.	State Estimation Errors (feet) – Case 5: Single Kalman Filter	270
Figure 116.	Non-Uniform Spacing for Modified Simpson's Rules	283

List Of Tables

	Page
Table 1. Uniform and Nonuniform Parameter Values	88
Table 2. Example Probability Values	110
Table 3. Example Fraction Values	130
Table 4. Simulated Test Cases (Interference/Jamming Levels)	149
Table 5. Parameter Values Chosen via Sheldon	156
Table 6. Design Parameters for Density Algorithm	178
Table 7. Chi-Squared Probability Values	239
Table 8. Search Criteria	240
Table 9. Example Combinations of Simpson's Rule	286
Table 10. Reduced-Order System Model States	292
Table 11. Elements of the Dynamics Submatrix $\mathbf{F}_{(red)11}$	293
Table 12. Elements of the Dynamics Submatrix $\mathbf{F}_{(red)12}$	294
Table 13. Elements of the Dynamics Submatrix $\mathbf{F}_{(red)22}$	294
Table 14. Elements of Process Noise Submatrix $\mathbf{Q}_{(red)11}$	294
Table 15. Elements of Process Noise Submatrix $\mathbf{Q}_{(red)22}$	294
Table 16. MMAE Blended Parameter Estimation Measure \hat{e}_{RMS}^a	295
Table 17. MMAE Blended State Estimation Measure \hat{e}_{RMS}^x - State 1	296
Table 18. MMAE Blended State Estimation Measure \hat{e}_{RMS}^x - State 2	296

Table 19.	MMAE Blended State Estimation Measure \hat{e}_{RMS}^x — State 3	296
Table 20.	M ³ AE Final State Estimation Measure \hat{e}_{RMS}^x — State 1	297
Table 21.	M ³ AE Final State Estimation Measure \hat{e}_{RMS}^x — State 2	297
Table 22.	M ³ AE Final State Estimation Measure \hat{e}_{RMS}^x — State 3	297

Abstract

The focus of this research is to provide methods for generating precise parameter estimates in the face of potentially significant parameter variations such as system component failures. The standard Multiple Model Adaptive Estimation (MMAE) algorithm uses a bank of Kalman filters, each based on a different model of the system. A new moving-bank MMAE algorithm is developed based on exploitation of the density data available from the residuals of the Kalman filters within the MMAE. The methods used to exploit this information include various measures of the density data and a decision-making logic used to move, expand, and contract the MMAE bank of filters. Parameter discretization within the MMAE refers to selection of the parameter values assumed by the elemental Kalman filters. A new parameter discretization method is developed based on the probabilities associated with the generalized Chi-Squared random variables formed by residual information from the elemental Kalman filters within the MMAE. Modifications to an existing discretization method are also presented, permitting application of this method in real time and to nonlinear system models or linear/linearized models that are unstable or astable. These new algorithms are validated through computer simulation of an aircraft navigation system subjected to interference/jamming while attempting a successful precision landing of the aircraft.

New Algorithms for Moving-Bank Multiple Model Adaptive Estimation

Chapter 1 - Introduction

1.1 Overview

Parameter and state estimation are critical in many of today's complex systems. A motivating example is that of providing an accurate navigation solution to an aircraft performing a precision landing. Sufficient accuracy of the state estimate is needed to provide the desired navigation solution. Adequate parameter estimation is needed so the system can adapt to a failure, such as the onset of interference or jamming of the onboard GPS receiver, and ensure performance is maintained.

The terms *failure detection and isolation (FDI)* and *parameter estimation* will appear synonymous in this document, since the methods and theories presented could be applied to either a failure detection or parameter estimation problem. If failure detection is viewed as event detection and event detection is accomplished by discerning changes in the parameter estimates, then adequate parameter estimation will lead to adequate failure detection.

The focus of this research is to develop methods for generating precise parameter estimates in the face of potentially significant parameter variations such as system component failures. The discrete-time system is presented in state space form with uncertain parameters (such as a failure status parameter) affecting the system matrices. This leads to the idea that nominal parameter values represent the system in its fully functional mode and variations in certain parameters indicate a possible failure or significant event. The objective is to identify the unknown parameter, \mathbf{a} , precisely given the time history of a set of measurements, \mathbf{z} . This is illustrated in Figure 1 where the measurements are used to generate the parameter estimate, $\hat{\mathbf{a}}$, and the state estimate, $\hat{\mathbf{x}}$, simultaneously.

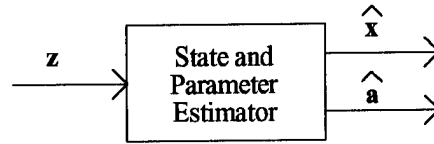


Figure 1. Generic State and Parameter Estimator

A special class of the generic state and parameter estimator is the Multiple Model Adaptive Estimator (MMAE) shown in Figure 2, with details presented in Section 2.2. The parameter estimate, $\hat{\mathbf{a}}$, is generated in the same manner as $\hat{\mathbf{x}}$ but is not shown to avoid cluttering the diagram. Each of J ($j = 1, 2, \dots, J$) Kalman filters in the bank models the operation of the system under a distinct failure mode, or more generally, under the conditions of a hypothesized discrete value of the vector of parameters that describe the system model. Each filter independently produces its own state estimate, $\hat{\mathbf{x}}_j$, and residual vector, \mathbf{r}_j . The residuals are monitored to determine which filter best models the system and its sensors at the current time. Each filter's state estimate, $\hat{\mathbf{x}}_j$, is blended together through a probability weighted average based on the conditional probability, p_j , of that filter modeling the true current system operating condition. This blending allows for partial failures being handled with hypothesized failure conditions composed of only "fully functional system" and "full failed sensor" conditions. A high probability of almost one indicates that a filter is extremely accurate in its modeling and will almost completely determine the final blended estimate, while all other model estimates will receive almost zero weighting. From an FDI standpoint, monitoring the elemental filter residuals or the conditional probabilities will provide insight into the failure mode of the system. Additionally, it will be shown in Section 2.2 that the parameter estimate $\hat{\mathbf{a}}$ can be directly found via Equation (51) as a best estimate of the current failure status.

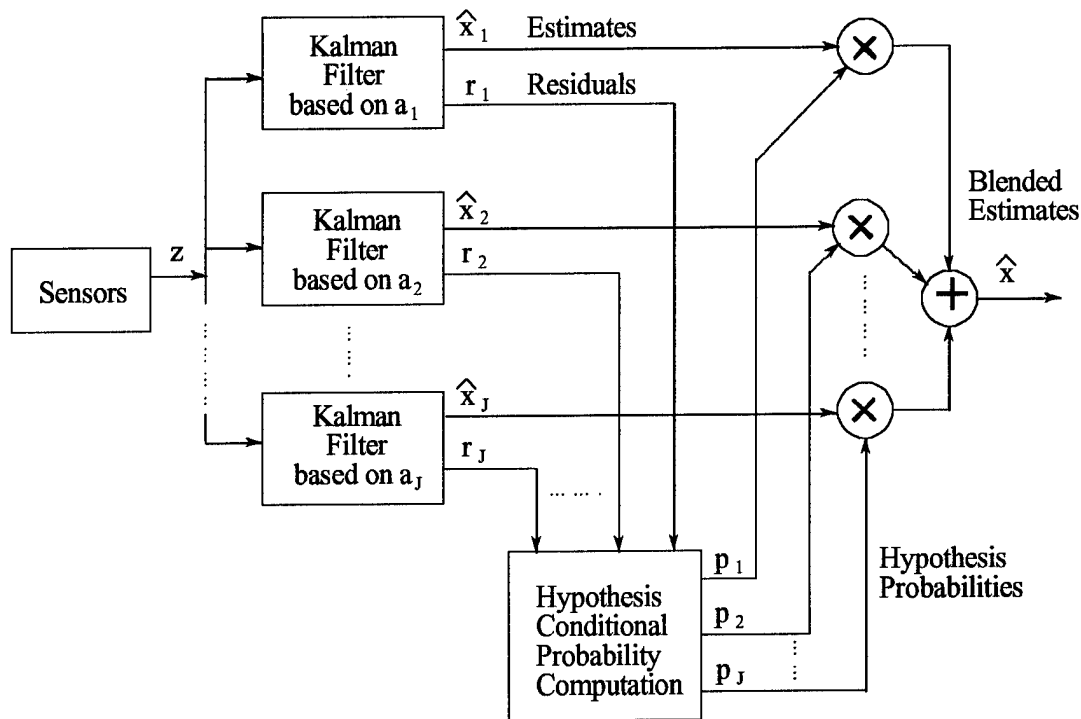


Figure 2. Multiple Model Adaptive Estimation Algorithm

The primary motivation for pursuing multiple model techniques such as MMAE over algorithms involving only one model of the system is the responsiveness of the MMAE to changes in parameter values. Classical state and parameter estimation algorithms, such as maximum likelihood estimators [64, 74, 75, 79, 80], that have only a single Kalman-like state estimator into which estimated parameters are inserted, do not have the capacity to respond as quickly or as well to real-world parameter changes as do multiple model algorithms incorporating parallel Kalman filters. It is more difficult and time-consuming to generate updated parameter estimates based on a single filter's residuals than to do so by comparing the quality of residuals from numerous filters, each based on a different hypothesized parameter value. One argument against multiple model algorithms was the computational loading associated with running several models in parallel. However,

advancements in computer technology (particularly distributed and parallel computing capability) have made MMAE's realizable for a large class of problems.

To avoid the potentially large number of elemental filters needed for an MMAE bank, the concept of a "moving bank" of fewer filters has been developed [18–20, 47, 66, 67]. For instance, if there are two uncertain parameters and each can assume 10 possible values, then $J = 10^2 = 100$ separate filters must be implemented in a full-scale fixed-bank MMAE, even if the parameters are treated as unknown constants. The moving-bank MMAE is identical to the full-bank estimator discussed previously, except J corresponds to the smaller number of elemental filters in the moving bank rather than the total number of possible discrete parameter vector values. There is then an on-line dynamic redeclaration of which points in the parameter space are to be used for the basis of the elemental filters within the MMAE, i.e., which points are to define the current "moving bank". In the example above, one might choose the three discrete values of each scalar parameter that most closely surround the estimated value, requiring $J = 3^2 = 9$ separate elemental filters, rather than 100. This is depicted in Figure 3.

Which particular J filters are in the bank at a given time can be determined by one of the five *ad hoc* decision mechanisms presented in Section 2.2.3, with the intention of keeping the estimate of the parameter in the bounds of the bank and "optimally" placed. For some of the decision mechanisms, the parameter estimate is kept in the center of the bank of filters. If the parameter estimate is found to move, then the bank of filters will move within the parameter space as shown in Figure 4, thus tracking the parameter.

For some of the decision mechanisms, the true parameter value may appear to have moved outside the bounds of the current bank; so the bank could expand to the coarsest level of discretization to bring the true parameter value into the bank, as illustrated in Figure 5. Similarly, once the true parameter value is deemed to lie within the portion of the parameter space currently spanned by the

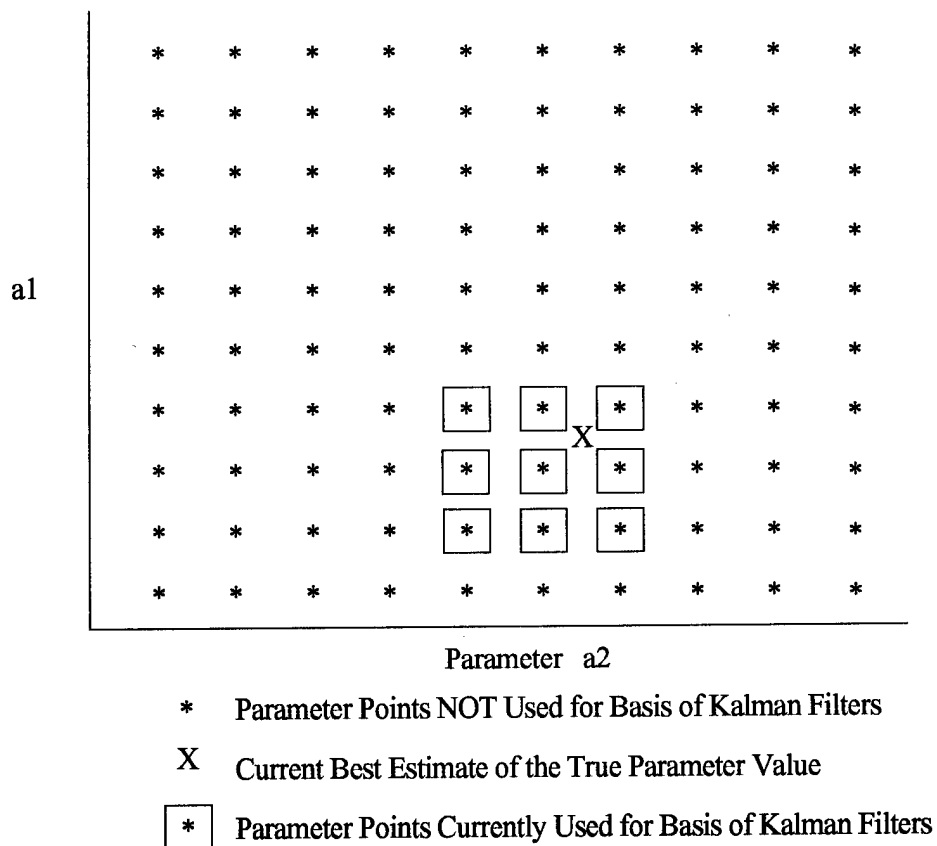


Figure 3. Moving-Bank MMAE for a Two-Dimensional Parameter Space

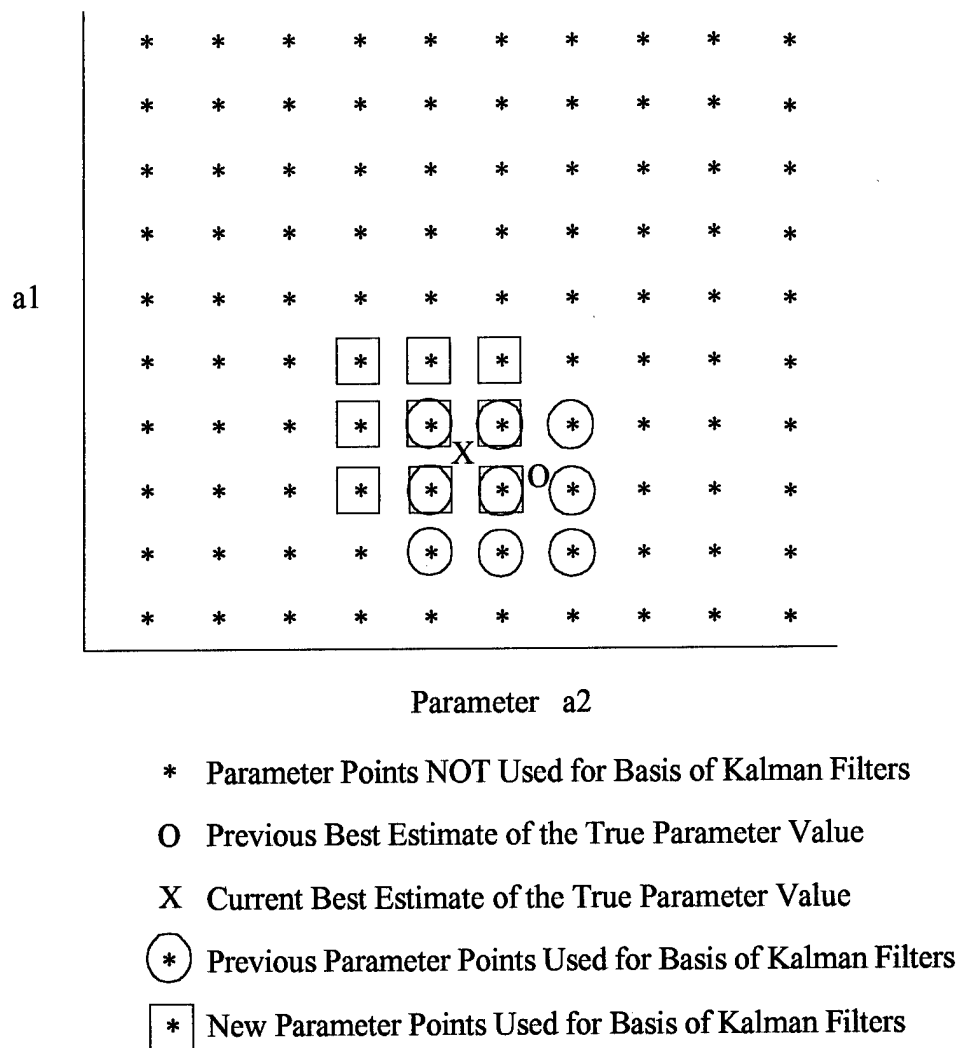


Figure 4. *Moved* Moving-Bank MMAE

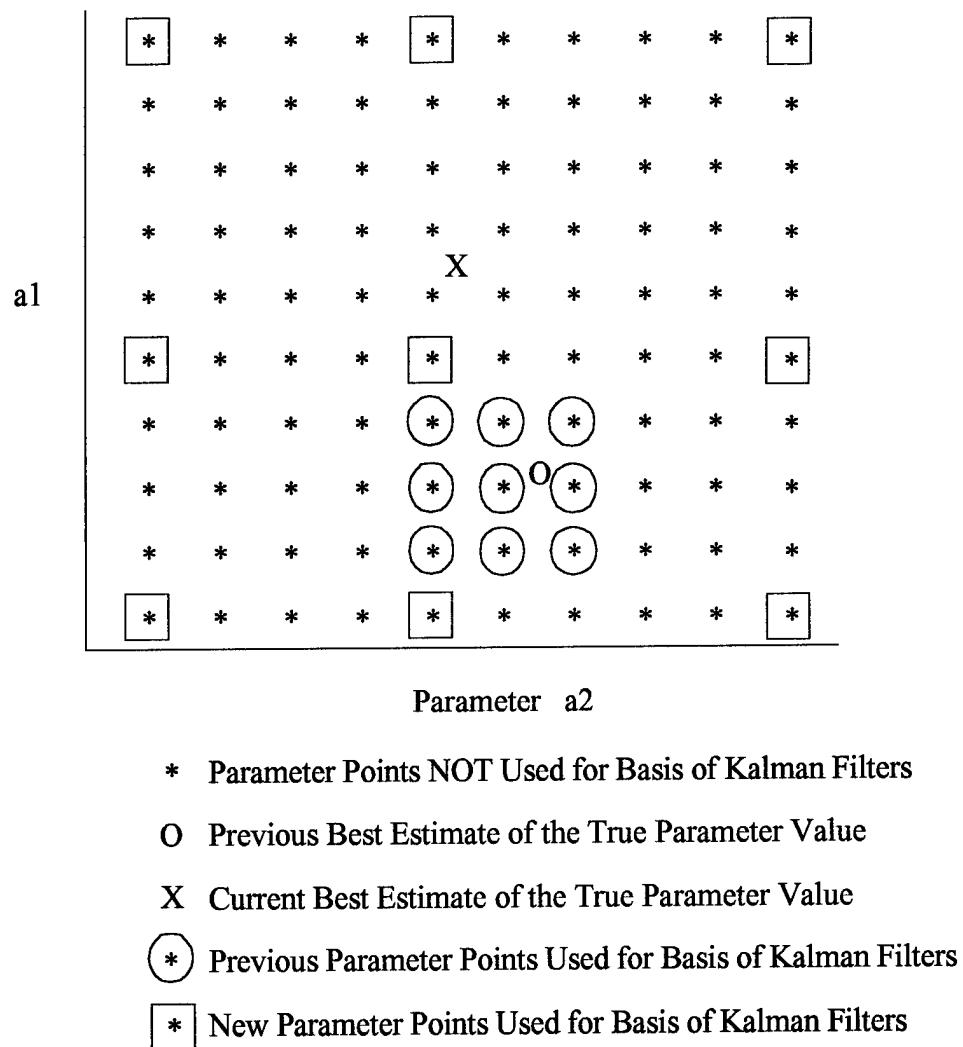


Figure 5. *Expanded* Moving-Bank MMAE

bank, and if that bank were not at the finest discretization level, the bank could be contracted to bring the filter-assumed values closer to the true parameter and thus improve the parameter (and state) estimate. In addition to the decision mechanisms introduced above, is the issue of how to discretize the parameter space spanned by the MMAE bank. This is addressing the question “If allowed J discretized points (where J is preselected) in the parameter space, where should they be placed for best state estimation precision, or best parameter estimation accuracy, or best performance with respect to some other chosen criterion?” One discretization method developed by Sheldon [68, 69] is presented in Section 2.2.3.2.

Given that MMAE is chosen as the multiple model algorithm being pursued in this research, Figure 1 is represented functionally by the *conventional* MMAE shown in Figure 6. Notice that the

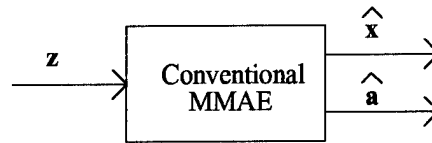


Figure 6. Conventional MMAE

parameters and states are still estimated simultaneously given the measurement history. However, optimal state estimation and optimal parameter estimation often require different choices for design variables such as tuning values for a Kalman filter. Additionally, state estimators based on system models are sensitive to the accuracy in the model parameters. This presents the designer with a trade-off between obtaining good state estimates versus precise parameter estimates. Therefore, a decomposition approach depicted in Figure 7 has been proposed by Miller [53] to give the designer the flexibility to optimize each estimator for its intended use. The architecture is referred to as a Modified MMAE or M^3 AE. The idea is first to perform the failure detection by tuning and discre-

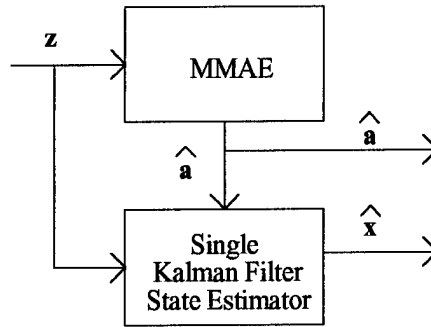


Figure 7. M³AE Architecture

tizing the MMAE bank for best parameter estimation and then to generate the state estimates with a single Kalman filter based on that good parameter estimate. This architecture has been shown by Miller [53] to provide superior state estimation results when compared to a conventional MMAE.

A variety of previously developed techniques used for parameter and state estimation will be presented in Chapter 2 as background information. Specifically, Fixed-Bank and Moving-Bank Multiple Model Adaptive Estimation algorithms have been used for parameter and state estimation [1, 2, 5, 14, 15, 17–22, 26, 27, 33–36, 42, 47, 48, 50–52, 55, 66–72]. The details of these algorithms will be presented in Section 2.2. Previous researchers have resorted to using *ad hoc* techniques within the MMAE algorithm to enhance performance. However, these *ad hoc* techniques are often suboptimal and could potentially be improved with closed form analytical solutions designed for optimality.

The major contributions of this dissertation are the development of new adaptive algorithms and discretization methods used to control the placement in parameter space of the discrete parameter values upon which to base the Kalman filters in a moving-bank MMAE. These algorithms and methods are applicable to either the conventional MMAE or the M³AE architecture. Computer

simulations were conducted for validation and proof of concept. The specific real-world application that motivated this research is parameter estimation for fault tolerant aircraft precision landing.

1.2 Dissertation Overview

Chapter 2 presents background concepts related to this dissertation. Discussions focus on existing theories pertaining to hypothesis testing and multiple model state and/or parameter estimation. Considerable emphasis is placed on moving-bank MMAE.

Chapter 3 gives a detailed account of the theories developed through this dissertation research. Two new algorithms dedicated to moving-bank MMAE are introduced and modifications to existing algorithms are presented. Various combinations of these algorithms are described with results discussed in Chapter 5. An attempt is made to explain the thought processes fully that led to the development of the algorithms rather than simply stating the final results.

Chapter 4 describes the example problem used to validate the theories presented in Chapter 3. Many of the algorithms have design parameters which are identified for the example problem. Simulation data based on the aircraft precision landing application is plotted and tabulated, to demonstrate the enhancement of performance accomplished through the new moving-bank MMAE methodologies.

Finally, Chapter 5 compares the results obtained using the various algorithms, draws pertinent conclusions and provides recommendations for future research.

Chapter 2 - Background Concepts

This chapter provides background theory for many of the techniques used in parameter and state estimation. Some emphasis is given to applying these techniques to an integrated, multi-sensor, aircraft navigation system. This will provide a motivating real-world problem, and the techniques are evaluated based on their ability to meet the more general research objectives of optimal parameter and state estimation.

A typical multi-sensor navigation system which incorporates an inertial navigation system (INS), a global positioning system (GPS) receiver, a barometric altimeter and a radar altimeter is schematically shown in Figure 8. This integration scheme is used as an example for the purpose of discussing the more general FDI techniques that apply to a larger class of problems. Depending on the application, consideration should be given to both differential GPS (DGPS) and carrier-phase GPS to enhance the accuracy of the navigation solution. Chapter 1 indicated that application of this research to aircraft precision landing is intended, and it is important to recognize the need for the radar altimeter to enhance the vertical-channel navigation precision during approach and landing maneuvers [7, 16].

A centralized Kalman filter (or possibly a distributed Kalman filter discussed in Section 2.1.5) is shown to integrate the various sensors and provide a feedforward correction to the navigation solution provided by the INS. An alternative to feedforward correction would be to feed back a correction to reset the INS and thus account for its errors. This approach is rarely used because the Federal Aviation Association (FAA) has deemed that feedback to the INS in these types of integration schemes has the potential to corrupt the INS and degrade the navigation solution seriously. Given a failure in any of the sensors, the navigation solution will degrade [30, 58, 75]. It is desired to detect

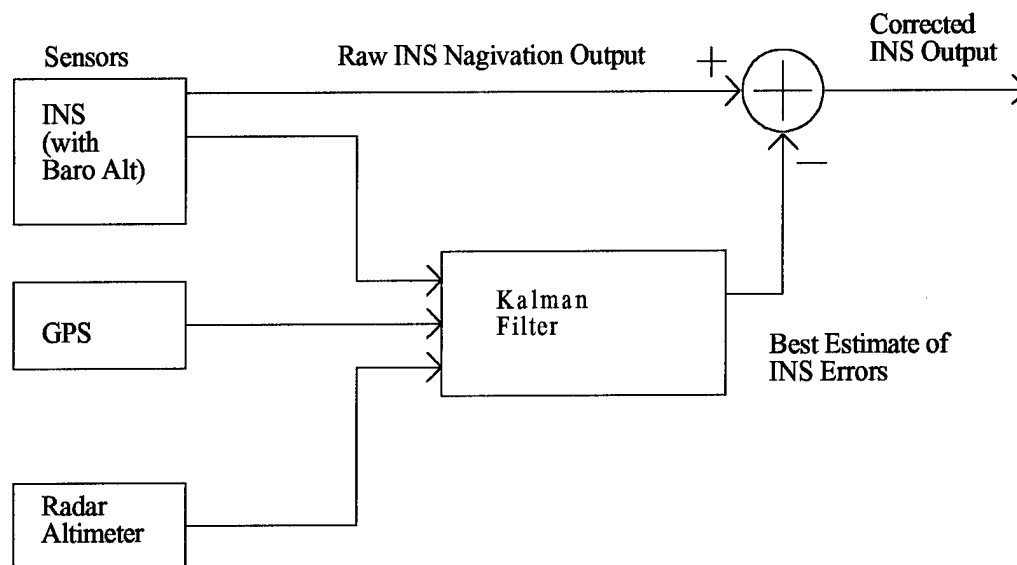


Figure 8. Multi-Sensor Navigation System

any such failures, or significant events such as the onset of interference or attempted jamming of GPS signals (not outright failures, but significant changes potentially causing unacceptable degradation of performance), and, when possible, compensate for these changes.

2.1 Parameter Estimation and Failure Detection

2.1.1 Voting Method

One of the simplest failure or event detection techniques is the use of redundant elements for voting. Given a system with triple redundancy, an algorithm can be easily designed that will compare the outputs of each identical element, allowing them to vote on the failure status condition of the redundant elements. Simply stated, if two of the elements agree but the third element provides

a significantly different reading, the latter is considered inadequate to provide accurate information and is removed from the system. Once this element is removed from the system, the algorithm is unable to isolate a failure. If the two remaining elements disagree, a failure has been detected but isolation is not possible. The major disadvantage of the voting method is the need for sufficient redundant hardware to ensure satisfactory performance [13, 75, 79], with the ensuing weight, volume, and cost disadvantages.

2.1.2 Chi-Square Test and the Kalman Filter

The chi-square test provides binary testing of simple hypotheses based on the Kalman filter residuals, $r(t_i)$. To set the context, details of the Kalman filter will be discussed first. A linear discrete-time Kalman filter with sampled data measurements is presented as the standard form considered for this research. Note that nonlinear models resulting in the use of extended Kalman filters (EKF) may be necessary to represent the true system adequately; however, the following development will focus on the linear Kalman filter for simplicity. Moreover, the system is actually defined by continuous-time dynamics equations with sampled data measurements but the “equivalent discrete-time model” [44, pp. 42–43] will be used for implementation on a digital computer. The state and measurement equations are as follows:

$$\mathbf{x}(t_i) = \Phi(t_i, t_{i-1})\mathbf{x}(t_{i-1}) + \mathbf{B}_d(t_{i-1})\mathbf{u}(t_{i-1}) + \mathbf{G}_d(t_{i-1})\mathbf{w}_d(t_{i-1}) \quad (1)$$

$$\mathbf{z}(t_i) = \mathbf{H}(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) \quad (2)$$

where:

- \mathbf{x} = n-dimensional system state vector
- Φ = state transition matrix, the discrete equivalent
of the system dynamics matrix
- \mathbf{B}_d = discrete equivalent of the system control input matrix
- \mathbf{u} = deterministic control input vector
- \mathbf{G}_d = discrete equivalent of the noise input matrix
- \mathbf{w}_d = discrete-time zero-mean white Gaussian dynamics
driving noise vector with covariance $\mathbf{Q}_d(t_i)$ at each t_i
- \mathbf{z} = m-dimensional measurement vector
- \mathbf{H} = system output matrix
- \mathbf{v} = discrete-time zero-mean white Gaussian measurement
noise vector with covariance $\mathbf{R}(t_i)$ at t_i

and initial condition $\mathbf{x}(t_0)$ modeled as Gaussian, with mean $\hat{\mathbf{x}}(t_0)$ and covariance $\mathbf{P}(t_0)$, and assumed independent of \mathbf{w}_d and \mathbf{v} . Also assume that \mathbf{w}_d and \mathbf{v} are independent. From the initial conditions, the Kalman filter states can be propagated using the following equations:

$$\hat{\mathbf{x}}(t_i^-) = \Phi(t_i, t_{i-1})\hat{\mathbf{x}}(t_{i-1}^+) + \mathbf{B}_d(t_{i-1})\mathbf{u}(t_{i-1}) \quad (3)$$

$$\mathbf{P}(t_i^-) = \Phi(t_i, t_{i-1})\mathbf{P}(t_{i-1}^+)\Phi^T(t_i, t_{i-1}) + \mathbf{G}_d(t_{i-1})\mathbf{Q}_d(t_{i-1})\mathbf{G}_d^T(t_{i-1}) \quad (4)$$

with measurement updates incorporated as follows:

$$\mathbf{A}(t_i) = \mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i) \quad (5)$$

$$\mathbf{K}(t_i) = \mathbf{P}(t_i^-)\mathbf{H}^T(t_i)\mathbf{A}^{-1}(t_i) \quad (6)$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)[\mathbf{z}_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)] \quad (7)$$

$$\mathbf{P}(t_i^+) = \mathbf{P}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}(t_i)\mathbf{P}(t_i^-) \quad (8)$$

The residual covariance, $\mathbf{A}(t_i)$, has been identified in Equation (5) and the Kalman filter residual vector is given by

$$\mathbf{r}(t_i) = \mathbf{z}_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-) \quad (9)$$

If the filter model matches the “truth” model, then the residuals will be a zero-mean white Gaussian process with known residual covariance, $\mathbf{A}(t_i)$ [44]. Larger magnitudes of the residuals than anticipated by the filter-computed $\mathbf{A}(t_i)$ indicate a mismatch between truth and the filter-assumed model. If the residuals have the anticipated characteristics for a period of time, but later increase in magnitude beyond what is anticipated, then this would imply that a parameter change has occurred in the real world. These increased magnitudes may appear as a nonzero mean or a change in the variance of the residuals. The chi-square random variable, $\chi(t_k)$, provides a test statistic that puts a quadratic penalty on variations in the residuals:

$$\chi(t_k) = \sum_{i=k-N+1}^k \mathbf{r}^T(t_i)\mathbf{A}^{-1}(t_i)\mathbf{r}(t_i) \quad (10)$$

where N is the size of a window sliding across the residual values (used to make decisions based on the most recent N residuals). The residual values are a result of the real-world dynamics and the filter-assumed dynamics model, but the system dynamics are not *explicitly* present in Equation (10); so this test is relying completely on the information contained in the residuals. A detection rule based on an empirically determined threshold, T , would be

$$\begin{aligned}\chi(t_k) > T &\Rightarrow \text{Parameter Change} \\ \chi(t_k) \leq T &\Rightarrow \text{No Parameter Change}\end{aligned}\tag{11}$$

The threshold value would be chosen to meet a desired performance specification such as minimized response time to real failures or parameter variations while maintaining no more than a maximum admissible false alarm rate. A false alarm is defined here as declaring a failure when no failure actually exists. Notice that the chi-square test combines the residual vector and the residual covariance into a single scalar, so it lacks the ability to monitor the individual scalar residuals. This prohibits the test from *isolating* failures, but it will often prove easy to implement and effective for *detecting* many types of failures. This shortcoming can be overcome by implementing MMAE algorithms which conceptually perform chi-square tests on many parallel Kalman filters. This approach will be presented in Section 2.2.

2.1.3 Sequential Probability Ratio Test (SPRT)

Another simple hypothesis test based on the Kalman filter residuals is the SPRT [76]. Recall that, if the filter model matches the “truth” model, then the residuals will be a zero-mean white Gaussian process with known residual covariance, $A(t_i)$. Associate this “no-fail” condition with hypothesis, H_0 . Assume that a failure has manifested itself as a change in the mean of a residual and the residual covariance remains unchanged. Associate this “failed” condition with hypothesis,

H_1 . Now form the log likelihood ratio for the i^{th} sample as

$$L_i = \ln \frac{f(\mathbf{r}(t_i)|H_1, \mathbf{Z}(t_{i-1}) = \mathbf{Z}_{i-1})}{f(\mathbf{r}(t_i)|H_0, \mathbf{Z}(t_{i-1}) = \mathbf{Z}_{i-1})} \quad (12)$$

where $f(\mathbf{r}(t_i)|H_j, \mathbf{Z}(t_{i-1}) = \mathbf{Z}_{i-1})$ is the density function of the residuals at time t_i conditioned on the hypothesis H_j for $j = 0, 1$ and the observed measurement history up to time t_{i-1} . Notationally, the measurement history random vector $\mathbf{Z}(t_i)$ is made up of partitions $\mathbf{z}(t_1), \dots, \mathbf{z}(t_i)$ as described in Equation (2) that are the measurement vectors available at the sample times t_1, \dots, t_i ; similarly, the realization \mathbf{Z}_i of the measurement history vector has partitions $\mathbf{z}_1, \dots, \mathbf{z}_i$. Then the test statistic, s_l , is given by

$$s_l = \sum_{i=1}^l L_i \quad (13)$$

This test statistic is compared to two thresholds, T_A and T_B , using the following decision logic

$$\begin{aligned} s_l &\geq \ln T_A \Rightarrow \text{Choose } H_1 \\ \ln T_B &< s_l < \ln T_A \Rightarrow \text{take another sample} \\ s_l &\leq \ln T_B \Rightarrow \text{Choose } H_0 \end{aligned} \quad (14)$$

The thresholds can be arbitrarily chosen based on simulations or determined analytically as shown below, unlike the chi-square test threshold which was only found empirically. Define a false alarm rate, α , as the probability of incorrectly declaring hypothesis H_1 is valid when H_0 is true. Define a missed alarm rate, β , as the probability of not declaring hypothesis H_1 is valid when H_1 is true.

The thresholds are then governed by

$$T_A \leq \frac{1 - \beta}{\alpha} \quad (15)$$

$$T_B \geq \frac{\beta}{1 - \alpha} \quad (16)$$

and in practice the thresholds are selected at the equalities. Hence, the thresholds are determined by the chosen values of α and β . The SPRT is very similar to the well known Neyman-Pearson test [65, pp. 107–110], which is the most powerful test given a simple hypothesis test satisfying a desired false alarm rate for a finite data set. The SPRT is superior [32] to the Neyman-Pearson test for sequential data sets such as those considered in this research.

The SPRT appears slightly more cumbersome than the chi-square test but provides a closed form solution for choosing threshold values. A single SPRT still lacks the ability to *isolate* failures, as did the chi-square test. However, multiple SPRT's could be used to test several hypotheses and thus improve parameter estimation [12].

2.1.4 Generalized Likelihood Ratio (GLR) Test

The GLR test is similar in nature to the Chi-Square test, but with the added benefit of failure detection and isolation. It is designed to distinguish between different types of failures and to estimate the magnitudes of the failure types [64, 74, 79, 80]. Like the Chi-Square test, the GLR test can exploit the residuals of a Kalman filter as its basis for failure analysis and FDI. The GLR test also does a threshold test, however, it compares a *generalized* likelihood ratio function, $l(t_i, \theta)$ (generated from the ratio of the log-likelihood of possible hypotheses analogous to Equation (12)), to a predetermined threshold to determine whether or not a failure has occurred. It is derived from the GLR equations described below [64, 75, 79, 80].

The hypotheses are established with a Kalman filter based on H_0 (no failure) and matched filters based on H_1 (a specific failure type added to the system). The ratio of the log-likelihood of the two hypotheses will be used to generate a *generalized* likelihood ratio function, $l(t_i, \theta)$, to be defined explicitly later in this section (see Equation (28)), which is used to declare failures via:

$$\begin{aligned} l(t_i, \theta) &> T &\Rightarrow & FAILURE \\ l(t_i, \theta) &\leq T &\Rightarrow & NO FAILURE \end{aligned} \quad (17)$$

If $l(t_i, \theta)$ is less than the predetermined threshold, T , then H_0 is declared true. Similarly, H_1 is declared true if $l(t_i, \theta)$ is greater than T . The parameter θ is the unknown time of the failure. The remainder of this section describes the derivation of the GLR algorithm for a single, step failure.

The following are the model equations upon which a Kalman filter might be based:

$$\mathbf{x}(t_i) = \Phi(t_i, t_{i-1})\mathbf{x}(t_{i-1}) + \mathbf{G}_d(t_{i-1})\mathbf{w}_d(t_{i-1}) \quad (18)$$

with discrete measurements described by:

$$\mathbf{z}(t_i) = \mathbf{H}(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) \quad (19)$$

The matched filters are designed for failure detection, not state estimation, and are based upon:

$$\mathbf{x}(t_i) = \Phi(t_i, t_{i-1})\mathbf{x}(t_{i-1}) + \mathbf{G}_d(t_{i-1})\mathbf{w}_d(t_{i-1}) \quad (20)$$

$$\mathbf{z}(t_i) = \mathbf{H}(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) + \mathbf{d}(t_i)n(t_i, \theta)\nu \quad (21)$$

where

$\mathbf{d}(t_i)$	=	failure vector
$n(t_i, \theta)$	=	failure function
ν	=	unknown size of the failure
θ	=	unknown time of the failure

Comparison of Equations (19) and (21) indicate that the matched filter characterizes failures by modeling them as variations in the actual measurements beyond the variations caused by the dynamics of the system or measurement noise, as indicated by the failure offset term, $\mathbf{d}(t_i)n(t_i, \theta)\nu$. Although the failure is modeled as a bias on the measurement, this model can also represent changes in the states caused by real world anomalies. The failure function term, $n(t_i, \theta)$, indicates the time

of the failure onset, θ , within a predetermined sliding “window” of time, and the type of failure that has occurred; i.e., ramp offset, step offset, etc. A sliding window of predetermined length is used to avoid a growing set of hypotheses and slides in time to cover all the data collected (as the window slides in time, the oldest data in the window is discarded as the new data enters the filter, maintaining the same total amount of information). The ν term is the magnitude of the failure and can be estimated by the GLR algorithm and can also be used for corrective feedback. Note that after the corrective feedback is utilized, the Kalman filter would be based on hypothesis H_1 , and thereafter used as the basis for calculating $l(t_i, \theta)$. Furthermore, the matched filter should assume the original no-fail hypothesis, H_0 , giving the algorithm the flexibility to recognize that the failed condition no longer exists. The column vector, $\mathbf{d}(t_i)$, specifies which of the measurement signals has the failure. In general, the likelihood ratio function, $l(t_i, \theta)$, is based on maximum likelihood estimates of θ and ν . The goal of the GLR algorithm is to identify the failure signal by recognizing variations in the residuals from their normal operating values.

The Kalman filter residuals $\mathbf{r}(t_i)$ are defined as

$$\mathbf{r}(t_i) = \mathbf{z}(t_i) - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-) \quad (22)$$

and the residuals for each hypothesis are described by

$$\begin{aligned} H_0 &: \mathbf{r}(t_i) = \mathbf{r}^0(t_i) \\ H_1 &: \mathbf{r}(t_i) = \mathbf{r}^0(t_i) + \mathbf{g}(t_i, \theta)\nu \end{aligned} \quad (23)$$

When the system is operating under normal conditions, the Kalman filter tracks the true states, and $\mathbf{r}^0(t_i)$ is zero-mean white Gaussian noise with covariance $\mathbf{A}(t_i) = \mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}(t_i)^T + \mathbf{R}(t_i)$.

When a failure occurs, a signal of unknown magnitude, $\mathbf{g}(t_i, \theta)\nu$, will be present in the residuals.

This signal is the failure residual offset and found through

$$\mathbf{g}(t_i, \theta) = \mathbf{H}(t_i)\mathbf{f}(t_i, \theta) + \mathbf{d}(t_i)n(t_i, \theta) \quad (24)$$

where the recursive failure quantity $\mathbf{f}(t_i, \theta)$ is given by

$$\mathbf{f}(t_{i+1}, \theta) = \Phi(t_{i+1}, t_i) [\mathbf{I} - \mathbf{K}(t_i) \mathbf{H}(t_i)] \mathbf{f}(t_i, \theta) - \Phi(t_{i+1}, t_i) \mathbf{K}(t_i) \mathbf{d}(t_i) n(t_i, \theta) \quad (25)$$

Note that the GLR algorithm is a function of the overall system behavior (Φ and \mathbf{H}) and Kalman filter gain \mathbf{K} as shown in Equations (24) and (25). If the failure is assumed to occur at the beginning of the sliding window discussed earlier, then Equations (24) and (25) can be simplified by setting $n(t_i, \theta) = 1$ for all t_i :

$$\mathbf{g}(t_i) = \mathbf{H}(t_i) \mathbf{f}(t_i) + \mathbf{d}(t_i) \quad (26)$$

$$\mathbf{f}(t_{i+1}) = \Phi(t_{i+1}, t_i) [\mathbf{I} - \mathbf{K}(t_i) \mathbf{H}(t_i)] \mathbf{f}(t_i) - \Phi(t_{i+1}, t_i) \mathbf{K}(t_i) \mathbf{d}(t_i) \quad (27)$$

The primary reason for this simplification is to reduce the computational burden associated with calculating several GLRs based on different values of θ . However, the consequence of this simplification is a delay in detecting the failure caused by waiting for the failure to reach the beginning of the sliding window. The Kalman filter outputs combined with the matched filter model determines the magnitude of the *generalized* likelihood ratio function defined as

$$l(t_i, \theta) = \frac{S^2(t_i, \theta)}{C(t_i, \theta)} \quad (28)$$

where

$$S(t_i, \theta) = \sum_{j=1}^i \mathbf{g}^T(t_j, \theta) \mathbf{A}^{-1}(t_j) \mathbf{r}(t_j) \quad (29)$$

is essentially the correlation of the observed residuals with the abrupt change signatures given by $\mathbf{g}(t_i, \theta)$ for the different hypothesized types and times of occurrence. Furthermore,

$$C(t_i, \theta) = \sum_{j=1}^i \mathbf{g}^T(t_j, \theta) \mathbf{A}^{-1}(t_j) \mathbf{g}(t_j, \theta) \quad (30)$$

is interpreted as the amount of information present in $\mathbf{z}(t_1), \dots, \mathbf{z}(t_i)$ when an abrupt change occurs at time θ . In both Equations (29) and (30), $\mathbf{A}(t_j)$ is given by

$$\mathbf{A}(t_j) = \mathbf{H}(t_j)\mathbf{P}(t_j^-)\mathbf{H}^T(t_j) + \mathbf{R}(t_j)$$

and the maximum likelihood estimate (MLE) of the unknown magnitude of the failure, ν , is found by

$$\hat{\nu}(t_i, \theta) = \frac{S(t_i, \theta)}{C(t_i, \theta)} \quad (31)$$

The residual covariance $\mathbf{A}(t_j)$ and the residuals are combined in Equations (24) and (25) or Equations (26) and (27) to form a linear combination of the residuals $S(t_i, \theta)$ and a deterministic value $C(t_i, \theta)$ defined in Equations (29) and (30). Finally, the decision rule given by Equation (17) is used to determine the system's failure condition.

Thus, the GLR algorithm involves a single Kalman filter, a matched filter, and the likelihood calculation. It determines failures by observing changes in the filter residuals and calculates the likelihood of each possible event by correlating the residuals with the corresponding failure signature. Figure 9 depicts an example of a multiple GLR test with a bank of matched filters designed for the no failure and step failure modes. The Kalman filter provides its residuals to the bank, wherein each filter is tuned to a certain type of failure mode. Each matched filter's output is tested against a hypothesis, H_k , corresponding to each hypothesized failure mode. An MLE is computed for each specific hypothesis. Each MLE is fed into the common test logic algorithm, similar to Equation (17), to determine the correct hypothesis.

One of the key benefits of the GLR test is the need for only one Kalman Filter. Additionally, only one matched filter is required for each failure type, since the algorithm estimates unknown variables, such as the magnitude of the failure type, in the FDI process. This is a great computational load benefit, especially in comparison to other multiple model techniques. Even though GLR has

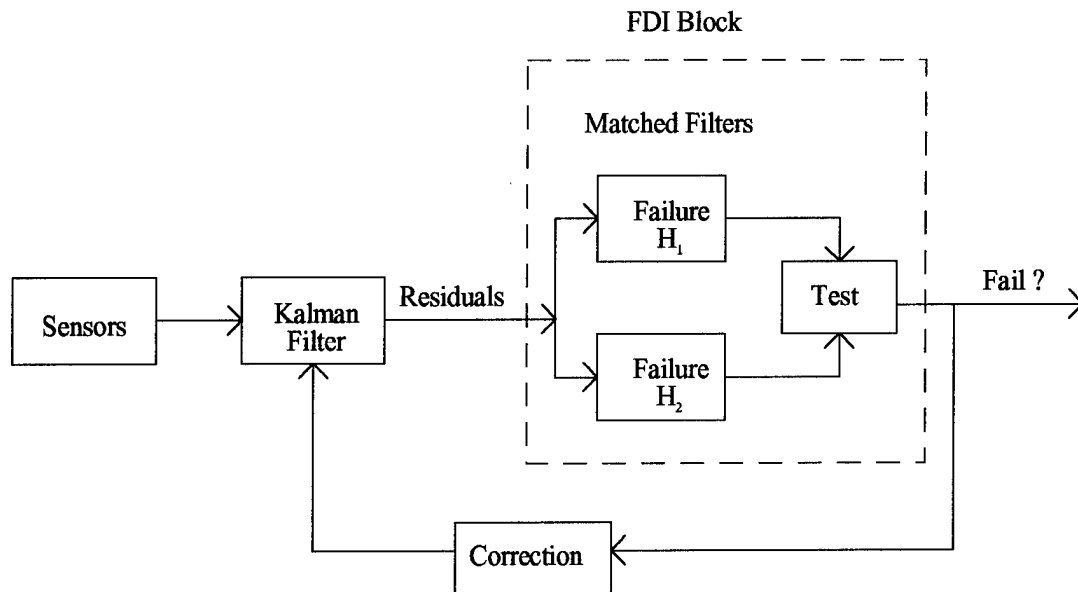


Figure 9. Multiple GLR Testing

been successfully applied to a wide variety of FDI applications [64, 74, 75, 79, 80], there are some limitations which include:

1. It has difficulty handling any parametric changes which may occur in the model due to changes in the system's operating condition. Thus, it is difficult for the GLR to model the dynamic nature of a system and its sensors to represent their behavior in the presence of failures. Therefore, typical GLR tests lack robustness since they are unable to detect parametric changes while looking for additive changes.
2. While detection of abrupt changes is a GLR's strength, detection of ramp failures is difficult. An unacceptable time delay may occur since the ramp-corrupted signal is slowly moving away from the desired signal and takes more time to cross the failure threshold [75]; thus, depending

on the length of the sliding window, the GLR test may fail to detect the failure. An additional drawback is that ‘windowing’ the estimate of θ leads to a direct reduction in the accuracy of the estimate of the size of the failure, ν .

3. Windowing may also cause an unacceptable delay in the identification of failures.

White [77,78] accomplished an indirect comparison of conventional MMAE-based techniques with a GLR/chi-square based technique applied to an FDI study performed by this author [75] on a GPS/INS based system. The following observations were made:

1. Both methods were effective at detecting interference failures (represented by increased measurement noise variances), but the GLR/chi-square based scheme suffered from much larger time delays as compared to the MMAE.
2. The GLR/chi-square scheme also experienced large time delays when returning to a nominal no-fail declaration after receiving a large amount of interference, as compared to the MMAE.
3. Additionally, the GLR/chi-square algorithm suffered from its inability to detect/identify ramp failures adequately.

In summary, the GLR/chi-square failure testing scheme experienced unacceptable time delays compared to the MMAE-based techniques, especially in the face of bias-like failures.

2.1.5 Distributed Kalman Filtering (DKF)

This section parallels the discussion first presented by Miller [53]. An alternative to the stand-alone or “centralized” Kalman filter is the distributed Kalman filter (DKF). The DKF developed by Carlson [8–10, 38], also called the federated filter, is another parallel structure like MMAE. However, unlike the MMAE, “local” filters in the DKF process *partitions* of the available measurement vector (whereas the “elemental” filters in the MMAE each process the *entire* measurement vector)

and then the information in the “local” filters is fused together in a “master” filter. DKF is based on simple and effective “information-sharing” methodology. The basic procedure involved in this information-sharing is [38]

1. Divide the total system information among several component filters or modules (“local” filters).
2. Perform local time propagation and measurement update processing (adding local sensor information as available).
3. Recombine the updated local information into a new total sum.

DKF has potential benefits gained through its modular structure, i.e., each component filter is managed and maintained independent of the others. DKF is useful in sensor failure scenarios because only the associated “local” filter might get corrupted by an undetected sensor failure, rather than all variables being corrupted as in a single centralized filter approach. However, DKF assumes statistical independence between the modules which is often impossible to validate.

2.1.6 Multiple Model Adaptive Estimation (MMAE)

The final parameter estimate technique for discussion is the use of multiple model adaptive estimation to represent a wide range of parameter variations within the system model. MMAE, like chi-square, SPRT, and GLR, exploits Kalman filter residual information, but it does so using multiple Kalman filters, each based on a specific hypothesis. The basic structure of an MMAE is shown in Figure 10 and the detailed equations governing this algorithm will be presented in Section 2.2. The major strength of MMAE is its ability to reconfigure rapidly in the presence of failures and thus provide accurate state estimates. However, an inherent trade-off exists between accurate parameter estimation or FDI, and accurate state estimation. Specifically, if one tunes the MMAE

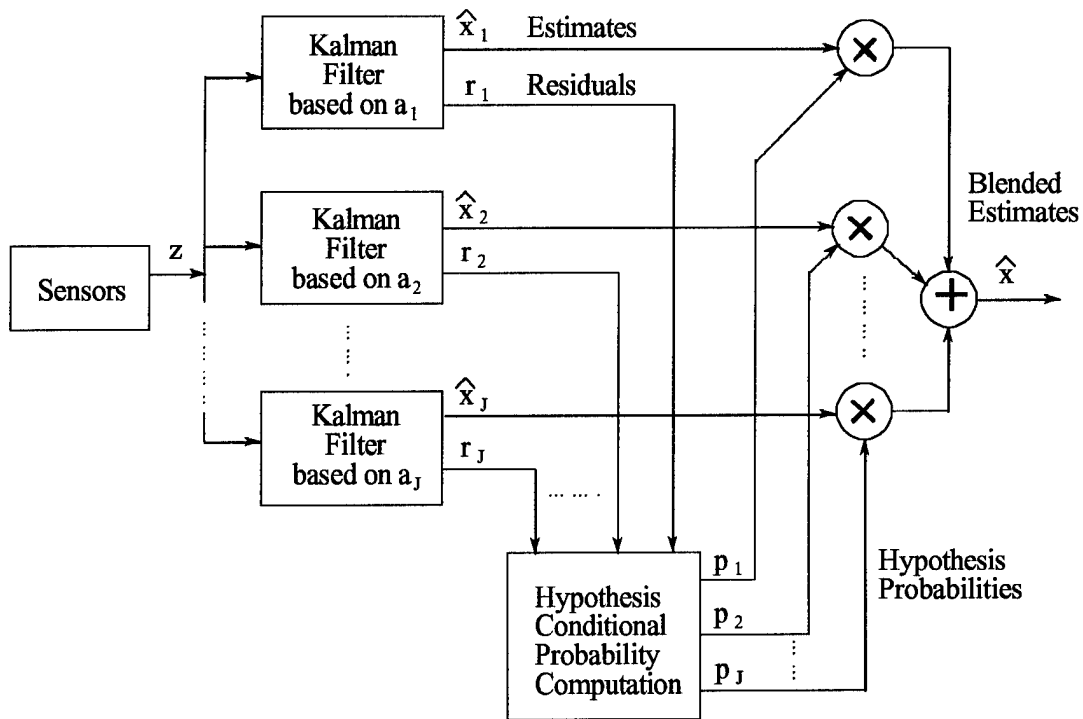


Figure 10. Multiple Model Adaptive Estimation Algorithm

bank for optimal parameter estimation, then these same tuning values will often result in less than optimal state estimation. Many researchers have focused on exploiting the capability of MMAE to provide state or parameter estimation separately as well as attempting to blend them ideally [1, 2, 5, 14, 15, 17–22, 26, 27, 33–36, 42, 44–48, 50–52, 55, 66–72]. By running multiple filters in parallel, residual information at each update is used to identify the system failure status and to reconfigure the system rapidly to failures. Unlike the GLR algorithm which implements only a single Kalman filter and multiple matched filters, the MMAE employs multiple Kalman filters to model the dynamic nature of the system (and its sensors) to represent performance in the presence of specific hypothesized failure conditions.

A logical choice for one is the no-fail condition, with the remainder of the filter bank comprised of filters based on hypothesized candidate failure types. The number of elemental filters will affect the granularity of the parameter estimation and ultimately the accuracy of the state estimation. Section 2.2 will address the various methods used to enhance MMAE performance, including the Moving-Bank MMAE, which is the focus of this research.

2.2 MMAE Review

This section presents the fundamentals of the MMAE algorithm in detail, along with various techniques that have been researched to enhance the performance of the MMAE concept. The focus will be on the Moving-Bank MMAE with some background presented on hierarchical structures and inter-residual distance feedback.

The following assumptions apply to the MMAE algorithm:

- The sampled-data system is adequately represented by a linear stochastic state differential model for a given parameter vector value, resulting in Gaussian probability density functions, and can be described equivalently by linear stochastic difference equations [45,47]. If nonlinear models are required to describe the system adequately, then extended Kalman filters would replace the linear Kalman filters in the MMAE structure and the associated probability density functions would be approximated as Gaussian. This simplifying assumption enables nonlinear modeling and approximate nonlinear filtering to fit into the context of the MMAE structure while recognizing the potential suboptimality in assuming Gaussian densities.
- The uncertain parameters to be estimated affect the system matrices (in the linear case, or structure-defining vector functions in the nonlinear case) or the statistics of the noises entering the system. This assumption covers a very broad class of problems since only the choice of measurement sources and choice of state variables are considered fixed.
- MMAE theory fundamentally assumes a discrete-valued parameter vector. In actuality, a parameter value typically varies over a continuous range of parameter space. Thus, parameter values will have to be discretized to some level of resolution for feasible implementation. Clearly, poor choices in discrete values for a continuous parameter would result in poor modeling by the MMAE elemental filters and thus poor estimation from the entire MMAE algorithm itself. This results because there might *not* be an elemental filter within the MMAE bank that has a good model of the system's current behavior. The choice of discretization is a major design issue being addressed by this research.

2.2.1 MMAE Fundamentals

The theory presented in this section will follow the development by Maybeck [45,47] closely. Let \mathbf{a} denote the p -dimensional vector of unknown parameters in the system model and assume that, in general, the range of \mathbf{a} is continuous. In most physically motivated problems, there is a finite range of reasonable parameter values, but within that finite range there are an uncountably infinite number of discrete point values for the parameter. A separate elemental filter within the MMAE will be associated with each discrete parameter value hypothesized during modeling. To make the number of filters in the bank finite and thus keep the problem tractable, the continuous range of \mathbf{a} is discretized into J representative values. More explicitly, let the model corresponding to \mathbf{a}_j (for $j = 1, 2, \dots, J$) be described by an “equivalent discrete-time model” for a continuous-time system with sampled-data measurements. This is similar to the development shown in Section 2.1.2 with the j^{th} filter model given by

$$\mathbf{x}_j(t_i) = \Phi_j(t_i, t_{i-1})\mathbf{x}_j(t_{i-1}) + \mathbf{B}_{dj}(t_{i-1})\mathbf{u}(t_{i-1}) + \mathbf{G}_{dj}(t_{i-1})\mathbf{w}_{dj}(t_{i-1}) \quad (32)$$

$$\mathbf{z}(t_i) = \mathbf{H}_j(t_i)\mathbf{x}_j(t_i) + \mathbf{v}_j(t_i) \quad (33)$$

$$E \{ \mathbf{w}_{dj}(t_{i-1})\mathbf{w}_{dj}^T(t_{i-1}) \} = \mathbf{Q}_{dj}(t_{i-1}) \quad (34)$$

$$E \{ \mathbf{v}_j(t_i)\mathbf{v}_j^T(t_i) \} = \mathbf{R}_j(t_i) \quad (35)$$

$$E \{ \mathbf{w}_{dj}(t_{i-1})\mathbf{v}_j^T(t_k) \} = \mathbf{0} \quad \forall i, k \quad (36)$$

Note that the parameter can affect Φ , \mathbf{B}_d , \mathbf{G}_d , \mathbf{H} , \mathbf{Q}_d and \mathbf{R} , as indicated by the subscript j on these matrices. Based on this model, the Kalman filter propagation and update equations are given by Equations (3) - (8) with the addition of the subscript j on all variables save \mathbf{z} and \mathbf{u} . More explicitly, the propagation equations are

$$\hat{\mathbf{x}}_j(t_i^-) = \Phi_j(t_i, t_{i-1})\hat{\mathbf{x}}_j(t_{i-1}^+) + \mathbf{B}_{dj}(t_{i-1})\mathbf{u}(t_{i-1}) \quad (37)$$

$$\mathbf{P}_j(t_i^-) = \Phi_j(t_i, t_{i-1})\mathbf{P}_j(t_{i-1}^+)\Phi_j^T(t_i, t_{i-1}) + \mathbf{G}_{dj}(t_{i-1})\mathbf{Q}_{dj}(t_{i-1})\mathbf{G}_{dj}^T(t_{i-1}) \quad (38)$$

and the update equations are

$$\mathbf{A}_j(t_i) = \mathbf{H}_j(t_i)\mathbf{P}_j(t_i^-)\mathbf{H}_j^T(t_i) + \mathbf{R}_j(t_i) \quad (39)$$

$$\mathbf{K}_j(t_i) = \mathbf{P}_j(t_i^-)\mathbf{H}_j^T(t_i)\mathbf{A}_j^{-1}(t_i) \quad (40)$$

$$\hat{\mathbf{x}}_j(t_i^+) = \hat{\mathbf{x}}_j(t_i^-) + \mathbf{K}_j(t_i)[\mathbf{z}_i - \mathbf{H}_j(t_i)\hat{\mathbf{x}}_j(t_i^-)] \quad (41)$$

$$\mathbf{P}_j(t_i^+) = \mathbf{P}_j(t_i^-) - \mathbf{K}_j(t_i)\mathbf{H}_j(t_i)\mathbf{P}_j(t_i^-) \quad (42)$$

The hypothesis conditional probability, $p_j(t_i)$, is defined as the probability that \mathbf{a} assumes the discrete value \mathbf{a}_j , conditioned on the observed measurement history to time t_i ,

$$p_j(t_i) = \text{Prob}[\mathbf{a} = \mathbf{a}_j | \mathbf{Z}(t_i) = \mathbf{Z}_i] \quad (43)$$

Then it can be shown [2, 29, 37, 45] that $p_j(t_i)$ can be evaluated recursively for all j via the iteration

$$p_j(t_i) = \frac{f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i | \mathbf{a}_j, \mathbf{Z}_{i-1})p_j(t_{i-1})}{\sum_{k=1}^J f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i | \mathbf{a}_k, \mathbf{Z}_{i-1})p_k(t_{i-1})} \quad (44)$$

in terms of the previous values of $p_1(t_{i-1}), \dots, p_J(t_{i-1})$ and the conditional densities for the current measurement $\mathbf{z}(t_i)$ to be defined explicitly in Equations (47)–(49).

Notationally, the measurement history $\mathbf{Z}(t_i)$ is made up of partitions $\mathbf{z}(t_1), \dots, \mathbf{z}(t_i)$ that are the measurement vectors available at the sample times t_1, \dots, t_i ; similarly, the realization \mathbf{Z}_i of the measurement history vector has partitions $\mathbf{z}_1, \dots, \mathbf{z}_i$. Furthermore, the Bayesian minimum mean square error (MMSE) estimate of the state is the probability-weighted average

$$\hat{\mathbf{x}}(t_i^+) = E\{\mathbf{x}(t_i) | \mathbf{Z}(t_i) = \mathbf{Z}_i\} = \sum_{j=1}^J \hat{\mathbf{x}}_j(t_i^+)p_j(t_i) \quad (45)$$

where $\hat{\mathbf{x}}_j(t_i^+)$ is the state estimate generated by a Kalman filter based on the assumption that the parameter vector equals \mathbf{a}_j .

Thus, the MMAE algorithm is composed of a bank of J separate Kalman filters, each based on a particular value $\mathbf{a}_1, \dots, \mathbf{a}_J$ of the parameter vector, as shown earlier in Figure 10. When the measurement \mathbf{z}_i becomes available at t_i , the residuals $\mathbf{r}_1(t_i), \dots, \mathbf{r}_J(t_i)$ are generated in the J filters, as shown in Equation (46):

$$\mathbf{r}_j(t_i) = \mathbf{z}_i - \mathbf{H}_j(t_i)\hat{\mathbf{x}}_j(t_i^-) \quad (46)$$

and used to compute $p_1(t_i), \dots, p_J(t_i)$ via Equation (44). Each numerator density function in Equation (44) is given by

$$f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathbf{Z}_{i-1}) = \beta_j \exp\{\cdot\} \quad (47)$$

where

$$\beta_j = \frac{1}{(2\pi)^{m/2} |\mathbf{A}_j(t_i)|^{1/2}} \quad (48)$$

and the expression in the brackets of Equation (47) is

$$\{\cdot\} = \left\{ -\frac{1}{2} \mathbf{r}_j^T(t_i) \mathbf{A}_j^{-1}(t_i) \mathbf{r}_j(t_i) \right\} \quad (49)$$

and where m is the measurement vector's dimension and $\mathbf{A}_j(t_i)$ is the residual covariance matrix at time t_i calculated in the j^{th} Kalman filter, as given by Equation (39). Note that the term β_j should not be confused with false alarm rates often represented as β . The denominator in Equation (44) is simply the sum of all the computed numerator terms and thus is the scale factor required to ensure that all $p_j(t_i)$ values sum to one.

One expects the residuals of the Kalman filter based on the best model to have the mean-squared value most in consonance with its own computed $\mathbf{A}_j(t_i)$, whereas "mismatched" filters have larger residuals than anticipated through $\mathbf{A}_j(t_i)$. Therefore, the filter based on the most correct assumed parameter value receives the most probability weighting. However, the performance

of the algorithm depends on there being significant differences in the characteristics of residuals in “correct” versus “mismatched” filters. Each filter should be tuned for best performance when the “true” values of the unknown parameters are identical to its assumed value for these parameters. One should specifically avoid the conservative philosophy of adding considerable dynamics pseudonoise, often used to open the bandwidth of a single Kalman filter to guard against divergence, because this tends to mask the differences between good and bad models. However, if this approach to tuning causes one of the filters to diverge (which is indicated by very large residuals), then it can be restarted with the current state estimate from the MMAE as computed from the nondivergent filters.

The Bayesian method was shown earlier to produce a state estimate (the conditional mean of \mathbf{x}) given by Equation (45) which will be denoted as

$$\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+) = \sum_{j=1}^J \hat{\mathbf{x}}_j(t_i^+) p_j(t_i) \quad (50)$$

The corresponding parameter estimate (the conditional mean of \mathbf{a}), can be generated as

$$\hat{\mathbf{a}}_{\text{MMAE}}(t_i) = \sum_{j=1}^J \mathbf{a}_j p_j(t_i) \quad (51)$$

Alternatively, the *maximum a posteriori* (MAP) method chooses the state estimate associated with the model having the highest probability which is given by

$$\hat{\mathbf{x}}_{\text{MAP-MMAE}}(t_i^+) = \hat{\mathbf{x}}_j(t_i^+) \text{ for } j \text{ such that } p_j(t_i) = \max_k [p_k(t_i)] \quad (52)$$

and similarly for $\hat{\mathbf{a}}_{\text{MAP-MMAE}}(t_i)$.

2.2.2 Simple Performance Enhancements

Several *ad hoc* techniques have been researched [14, 15, 42, 45, 47, 48, 50–52, 55, 70, 71] with the intent of enhancing the performance of the standard MMAE algorithm presented in the previous section. This section will briefly describe the following techniques in order.

1. Kalman Filter Tuning
2. β Dominance Compensation
3. Scalar Penalty Modification
4. Lower Bounding Conditional Probabilities
5. Probability Smoothing
6. Scalar Residual Monitoring
7. Increased Residual Propagation
8. Markov Process Modeling of Hypothesis Conditional Probabilities
9. Interacting Multiple Model Form of MMAE
10. Dithering

1. Kalman filter tuning refers to modifying the covariance of the process noise, \mathbf{Q}_{dj} , or the measurement noise, \mathbf{R}_j , of the elemental filters. As mentioned in Section 2.2.1, the addition of considerable process pseudonoise can deteriorate the detection capability of the algorithm and will often result in detection delays or missed alarms. However, appropriate tuning of these noise levels is often effective in reducing false alarms and/or missed alarms [14, 15, 22, 42, 45, 47]. Although this technique is primarily *ad hoc* since exact tuning values are not determined analytically, phys-

ical insights into the problem will often assist in determining the order of magnitude of the noise covariances. Specifically, one seeks tuning conditions such that $\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$ is approximately equal to m (the measurement dimension) when the hypothesized parameter value \mathbf{a}_j is a good match to the true parameter value, and significantly larger than m when the assumed parameter value is a poor match to truth.

2. Beta dominance is the tendency of the probability evaluator to calculate probabilities incorrectly because of the impact of the β_j term in Equation (48). If all the exponential terms, $\{\cdot\}$, in Equation (49) were approximately the same size for all the elemental filters, one would desire all elemental filters to be deemed equally adequate. However, Equations (44) and (49) would put the *highest* probability on the elemental filters with the *smallest* $|\mathbf{A}_j|$ value. This is inappropriate weighting since the size of $|\mathbf{A}_j|$ has *nothing* to do with the correctness of the failure detection. A typical representation of a sensor failure is to zero out the row of \mathbf{H}_j corresponding to the failed sensor. All other things being equal, the filters assuming this type of failure will tend to have smaller $|\mathbf{A}_j|$ values, and thus an MMAE used for sensor/actuator failure detection will be prone to false alarms on sensor failures. Two methods have been used to remove the β dominance effect [21, 47, 48, 50–52, 70, 71]. The first uses scalar residual monitoring, to be discussed presently. A second technique simply removes the β_j term in the conditional density resulting in a modified “density”

$$f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathbf{Z}_{i-1}) = \exp \left\{ -\frac{1}{2} \mathbf{r}_j^T(t_i) \mathbf{A}_j^{-1}(t_i) \mathbf{r}_j(t_i) \right\} \quad (53)$$

The algorithm functions properly with the β_j term removed because the denominator in Equation (44) is the sum of all possible numerators and will normalize the probabilities. Specifically, the

probabilities, p_j , will still sum to one even if the area under each of the modified “densities” is no longer unity. This research does *not* use this technique.

3. Increasing the magnitude of the negative scalar value of $-\frac{1}{2}$ in the $\{\cdot\}$ term in Equation (49) can decrease the probability convergence time for the MMAE. This technique is referred to as a scalar penalty increase since increasing the magnitude of the $-\frac{1}{2}$ term drives probabilities associated with large residuals to zero faster. Ideally, this should result in a faster convergence of the conditional probabilities; however, increasing the scalar penalty also increases fluctuations in the probabilities and thus the false alarm rate [48]. The “best” value of the scalar penalty is empirically determined through simulations. This research does *not* use this technique.

4. Placing a lower bound on the conditional probabilities, p_j , has been used to prevent filter lock-out and delays in failure identification [14, 15, 21, 42, 45, 47, 48, 50–52, 70, 71]. Due to the recursive nature of the probability calculation, Equation (44), if a probability is ever allowed to become zero, it will remain at zero thereafter. Likewise, if a failure occurred and the probability associated with the filter matching this failure hypothesis were allowed to get too small, it may take several iterations for this probability to increase and properly declare the failure. To prevent this from occurring, the probabilities are artificially bounded above zero to an empirically determined value and rescaled to ensure that they still sum to one. Alternatively, the IMM approach [3, 5] (and some others) discussed presently removes the lower bounding of the conditional probabilities and replaces such lower bounding with a Markov model for the time-propagation of the p_j values. However, this requires knowledge of the J-by-J probability state transition matrix for that Markov process model. This assumption will be discussed further under *Markov Process Modeling* that follows. Although lower bounding the conditional probabilities is an *ad hoc* technique, it accomplishes analogous behavior in the computed probabilities without detailed knowledge of the entire probability state transition matrix. In fact, the probability state transition matrix is set equal to the identity matrix in

the case of using lower bounds, indicating that the parameters are modeled essentially as constants over time. This research does apply lower bounding of the probabilities.

5. Immediately following a change in the system operating mode, the probabilities will go through a transient period before converging to the correct solution. Probability smoothing is used to minimize the momentary false alarms based on these transients [21, 48, 51, 52, 70, 71]. The probabilities are smoothed over a moving window, i.e., averaged over a number of data samples. The size of the window is empirically chosen, with a large window size causing detection delays and a small window size allowing more false alarms. This research does *not* use this technique.

6. Scalar residual monitoring is used to minimize false alarms for sensor type failures and involves component terms in the summation for the likelihood quotient defined by the following quadratic

$$L_j(t_i) = \mathbf{r}_j^T(t_i) \mathbf{A}_j^{-1}(t_i) \mathbf{r}_j(t_i) \quad (54)$$

This term comes directly from Equation (49) and, based on discussions up to this point, it is clear that the useful information pertaining to the “correctness” of the parameter values is captured by this quadratic [45]. Inspection of Equations (44), (49) and (54) reveals that the probability calculations are dependent on the magnitude of the *entire quadratic form*. However, given a sensor failure modeled by zeroing out the k^{th} row of \mathbf{H} corresponding to the failed sensor, the *predominant indicator* of that failure ought to be a large value of the single scalar term in the likelihood quotient associated with the k^{th} scalar residual [50], namely:

$$L_{j_k}(t_i) = \frac{r_{j_k}^2(t_i)}{A_{j_k}(t_i)} \quad (55)$$

in any of the elemental filters (such as the H_0 -based filter) except for the one specifically designed to expect such a failure. The term additional voter implies that scalar residual monitoring would be used in conjunction with the standard conditional probability hypothesis testing. The *ad hoc* nature of scalar residual monitoring is two-fold. First, an empirically chosen threshold is compared to the scalar likelihood quotient given by Equation (55). Second, the amount of weight given to the additional vote via scalar residual monitoring versus the standard MMAE hypothesis test is *ad hoc*. This research does *not* use this technique, however, the information contained in the likelihood quotient, Equation (54), will be exploited extensively.

7. Increased residual propagation is another method used to reduce the decision convergence time for the MMAE [21, 48]. The idea is to propagate the Kalman filter state estimates, without updating, for a few sample periods while still monitoring the residuals. This allows the residuals in the elemental filters based on incorrect hypotheses or assumed parameter values, to grow much larger since the usual measurement updates are no longer correcting the state estimates toward the actual measurements; such corrections tend to mask the impact of an incorrect hypothesis. The number of propagations allowed before an update must be determined empirically. The risk involved in implementing this technique is an increase in the fluctuations of the conditional probabilities and thus false alarms [21, 48]. Clearly, this would also degrade the precision of the state estimates. This research does *not* use this technique.

8. If the parameter vector \mathbf{a} can be shown to be a *Markov* process, then the conditional probabilities are propagated based on the *Markov* process development shown in [5, 45, 55]. Of particular interest is the probability state transition matrix $\mathbf{T}(t_i, t_{i-1})$ which allows propagation of the conditional probabilities in place of Equation (44). Let the elements of the J -dimensional vector $\mathbf{p}(t_i)$ be the probabilities $p_j(t_i)$ defined in Equation (43); then the following is the Markov model for prop-

agation:

$$\mathbf{p}(t_i) = \mathbf{T}(t_i, t_{i-1})\mathbf{p}(t_{i-1}) \quad (56)$$

The obvious problem that arises from this approach is the need to determine all the entries of the probability state transition matrix, which often requires considerable physical insight or some degree of arbitrary assignment. The benefits of this approach are the ability to implement the IMM approach [3, 5] discussed presently and the avoidance of lower bounding the conditional probabilities with *ad hoc* values to handle time-varying p_j 's. The predictive nature obtained by using Equation (56) can overcome the previous problem of lock-out, i.e., it will change a $p_j = 0$ to some small value prior to its need to grow large and force a change in the choice of hypothesis, if $\mathbf{T}(t_i, t_{i-1})$ is nondiagonal. Typically, choosing a good lower bound for the p_j 's is much easier than fully defining $\mathbf{T}(t_i, t_{i-1})$ with any level of certainty. This research does *not* use this technique.

9. A technique called Interacting Multiple Model (IMM) performs a restart of all the filters in the bank after every sample period [3, 5]. Given that the probability state transition matrix in the standard MMAE structure equals identity, the theory in [3, 5] dictates using $\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+)$ from Equation (50) or some other appropriate function of p_j and $\hat{\mathbf{x}}_j$ values from the current or most recent two sample periods to restart the filters. The IMM approach is intended to improve state estimation but will often degrade FDI performance since subtle failures will often go undetected. This results from the filters tracking the subtle failure and the intermixing of state estimates in all the elemental filters. In contrast, if the filters are not reset at each sample time, then the subtle failure will eventually cause the conditional probability, p_j , associated with this type of failure to increase and thus be detected. This research does *not* use this technique; however, initialization of filters with newly declared parameter values will be used as described in Section 3.1.5.2.

10. Dither, defined as the purposeful introduction of excitation into the system, is often necessary for any FDI algorithm, including MMAE, to detect and isolate failures adequately. The primary

reason is the need to excite the system away from a quiescent state that does not properly depict the true failure status. Otherwise, the residuals associated with the elemental filters that poorly match the true system will be indistinguishable from the residuals for the filters that correctly match truth. In some cases, the system will undergo sufficient dynamic maneuvers to induce the needed stimuli, i.e., several high-g turns for an aircraft. However, autonomous dithering will tend to enhance FDI in benign conditions, i.e., straight and level flight. A number of dither signals have been evaluated, including white noise, square waves, triangle waves, combinations of these forms, and sine waves [51, 52, 70, 71] with satisfactory results. The white noise dither (which is in reality only white over the bandpass of the system) provides excitation at all frequencies, whereas the periodic dithers must be implemented at carefully chosen frequencies and phasings, or else sweep across the frequency range of interest and dwell at discrete values. The periodic dither has the advantage that the residuals can be monitored not just for magnitude but also for frequency effects corresponding to the input dither. Specifically, the residuals associated with the wrong hypothesis will display the periodic signal at the known dither frequency, while the residuals for the correct hypothesis will be zero-mean white Gaussian process of covariance $\mathbf{A}_j(t_i)$, since the effect of purposeful controls on $\mathbf{z}(t_i)$ will be exactly compensated by the $[\mathbf{H}_j(t_i)\hat{\mathbf{x}}_j(t_i^-)]$ term in Equation (46) if \mathbf{a}_j corresponds to the correct parameter value. A negative side effect of autonomous dithering is constructive interference, which refers to the interaction between multiple dither signals due to coupling inherent in the system model. This coupling has been shown by [70, 71] to result potentially in a lower frequency dither than desired and thus a degradation of the frequency monitoring used for FDI. Simple variations in frequency and phase between channels of the system will often prevent this type of interference. Careful consideration must be given to keeping dither signals subliminal so as not to introduce unwanted dynamics into the system, i.e., avoid fatiguing the pilot of an aircraft with high-g pulsing. This research does *not* use this technique.

2.2.3 Moving-Bank MMAE

The moving-bank MMAE was motivated in Chapter 1 with brief descriptions of moving, expanding and contracting the bank. Additional research has focused on methods to discretize the parameter space in terms of choosing the best filter-assumed parameter values within the MMAE, as well as on decision logics to control the bank motion. Further details regarding these topics are presented here.

2.2.3.1 Decision Logics

The following section parallels the discussion first presented by Maybeck [47]. Five decision logics have been investigated by Maybeck and others [18–20, 45, 47, 66, 67] with the intention of keeping the estimate of the parameter in the bounds of the bank and “optimally” placed. They are:

1. Residual Monitoring
2. Parameter Position Estimate Monitoring
3. Parameter Position and “Velocity” Estimate Monitoring
4. Probability Monitoring
5. Parameter Estimation Error Covariance Monitoring

A brief summary of each follows.

Residual Monitoring. Recall the likelihood quotient given by Equation (54):

$$L_j(t_i) = \mathbf{r}_j^T(t_i) \mathbf{A}_j^{-1}(t_i) \mathbf{r}_j(t_i)$$

In the case of scalar measurements, this is the current residual squared, divided by the filter-computed variance for the residual. When the true parameter value does not lie in the current moving-bank region, all J likelihood quotients can be expected to exceed a threshold level T_L , the numerical value of which is set in an *ad hoc* manner during performance evaluations. Thus, a possible detection

logic would indicate that the bank should be moved and/or expanded at time t_i if

$$\min[L_1(t_i), L_2(t_i), \dots, L_j(t_i)] \geq T_L \quad (57)$$

Moreover, the elemental filter based on \mathbf{a}_j nearest to the true parameter value should have the smallest likelihood quotient, thereby giving an indication of the direction to move the bank. Tests analogous to Equation (57) can be applied to a subset of the current bank members, as those along one edge of a rectangular bank in a 2-dimensional parameter space, to indicate that the subset is not near the true parameter value. Although this logic would respond effectively to a real need to move or expand the bank, it would also be prone to false alarms induced by single large samples of measurement noise, since it is based on the *single* residual vector at time t_i .

Parameter Position Estimate Monitoring. Another means of keeping the true parameter value in the region bracketed by the moving bank is to keep the bank centered (as closely as possible in view of the discrete values that \mathbf{a} is allowed to assume) on the current estimate of the parameter. This estimate was shown in Equation (51) as

$$\hat{\mathbf{a}}(t_i) = E\{\mathbf{a} | \mathbf{Z}(t_i) = \mathbf{Z}_i\} = \sum_{j=1}^J \mathbf{a}_j p_j(t_i) \quad (58)$$

If the distance from the parameter value associated with the center of the bank to $\hat{\mathbf{a}}(t_i)$ becomes larger than some chosen threshold, a move of the bank in that direction is indicated. Since $\hat{\mathbf{a}}(t_i)$ depends on a *history* of measurements rather than just the single current measurement, it is less prone to the false alarms discussed in the previous method.

Parameter Position and "Velocity" Estimate Monitoring. If the true parameters are slowly varying, past values of $\hat{\mathbf{a}}(t_i)$ can be used to generate an estimate of parameter "velocity". This, along with the current position estimate $\hat{\mathbf{a}}(t_i)$, can be used to compute a predicted parameter position, one

sample period into the future. If the distance between the bank center and that projection exceeds some selected threshold, the bank can be moved in anticipation of the true parameter movement. This approach introduces lead to the moving-bank logic, but also a higher level of uncertainty and erratic bank movement.

Probability Monitoring. The conditional hypothesis probabilities, p_j , are another indication of the correctness of the parameter values \mathbf{a}_j assumed by the elemental filters of the current bank. If any of these rise above a chosen threshold level, the bank can be moved in the direction of the \mathbf{a}_j associated with the highest p_j . In this scheme, the bank seeks to center itself on the elemental filter with the highest conditional probability weighting. Again, since p_j depends on a history of measurements, this method should not be as sensitive to single bad samples of measurement corruption as is the case under residual monitoring.

Parameter Estimation Error Covariance Monitoring. This concept is discussed last because it has a somewhat different purpose than deciding when and in what direction to move the bank. It is also possible to change the size of the bank by altering the discretization level of the parameter space, as shown in Figure 5 on page 7. For example, initial acquisition can be enhanced by choosing the values $\mathbf{a}_1, \dots, \mathbf{a}_J$ so that they coarsely encompass all possible parameter values, rather than use a small bank and force it to seek a true parameter value that may well be outside the region of its assumed parameter values. Then, once a “good” parameter estimate has been achieved with this coarse discretization, the size of the bank can be contracted and the smaller bank centered on that good value. To help make such a contraction decision, it would be useful to monitor the parameter estimation error conditional covariance, computable [45] as

$$\begin{aligned}
\mathbf{P}_a(t_i) &= E\{[\mathbf{a} - \hat{\mathbf{a}}(t_i)][\mathbf{a} - \hat{\mathbf{a}}(t_i)]^T | \mathbf{Z}(t_i) = \mathbf{Z}_i\} \\
&= \sum_{j=1}^J [\mathbf{a}_j - \hat{\mathbf{a}}(t_i)][\mathbf{a}_j - \hat{\mathbf{a}}(t_i)]^T \cdot p_j(t_i)
\end{aligned} \tag{59}$$

When an appropriately chosen scalar function (norm) of this matrix falls below a selected threshold, the bank can be *contracted* about the parameter estimate. One such norm is the weighted sum of the diagonal terms of \mathbf{P}_a for a moving bank constrained to be a square region in a two-dimensional parameter space [47]. In general, one could use different discretization coarseness decisions in individual directions of the parameter space, allowing rectangular banks as well as squares.

An indication of the need to expand the size of the bank can be obtained from residual monitoring as before. If all likelihood quotients from Equation (54) are large and close in magnitude, then *none* of the current elemental filters appear to have a good model (hypothesized parameter value) and no clear indication of the true parameter's value is provided, and it is more appropriate to expand the bank than to attempt to move it. The error covariance could then be monitored for making the decision to return the bank to a smaller size. Since Equation (59) depends on the current choice of \mathbf{a}_j values, this error covariance is *not* a reliable indicator for the decision to expand; the $\mathbf{P}_a(t_i)$ so computed is artificially bounded from above by the current size of the bank.

Regardless of which technique listed above is used, when the bank is either moved or expanded, any filter corresponding to newly declared \mathbf{a}_j locations must be initialized with $\hat{\mathbf{x}}_j(t_i)$, $\mathbf{P}_j(t_i)$, and $p_j(t_i)$ values. A reasonable choice for $\hat{\mathbf{x}}_j(t_i)$ is the current moving-bank blended estimate $\hat{\mathbf{x}}(t_i^+)$. For $p_j(t_i)$, the total probability weight of one minus the sum of the unreset $p_j(t_i)$ values is divided among the new filters in the bank [23, 49]. Although this can be apportioned equally, better performance results when it is divided based on the relative correctness of the new filters being added [23,

49]. This correctness is determined from Equation (49) for each new filter and is used to divide the probability proportionately, i.e., the most correct filters will have a small value for the likelihood quotient given by (54) and thus receive the greatest probability allocated to the new filters.

Before a move or expansion, it may be desirable to warm up the new filters before bringing them on line, allowing initial transients in $\hat{x}_j(t_i)$ and $P_j(t_i)$ to die out. To accomplish this, the move threshold is left intact, and an additional, tighter warm-up threshold is used to indicate whether or not to warm up any filters. Until the original move threshold is surpassed, these new elemental filters do not affect the adaptive filter output.

2.2.3.2 *Sheldon Discretization*

This section represents a joint effort with Miller [53]. It is important to realize that the level of discretization of a continuous parameter space directly impacts (1) the ability of the filter bank to “surround” the parameter and (2) the average and maximum distance from the true parameter to the parameter value assumed by one of the elemental filters. It is critical that the parameter lie within the bank’s range of coverage for adequate parameter estimation, and it is desirable to have it close to one of the elemental filters for improved estimation. One *ad hoc* method for choosing the coarseness of discretization is to conduct a performance analysis of a single Kalman filter against a truth model and to vary one parameter of the truth model in one direction at a time, until the filter, based on the nominal parameter point, yields unacceptable levels of degraded performance [18–20, 45, 67–69]. Sheldon [68, 69] instead chose optimally discretized parameters by minimizing one of three cost functions, depending on design goals of good state estimation, parameter estimation, or state control. Consider the first case, in which the cost functional, C , represents the average value of the mean squared estimation error, where the average is taken as the true parameter ranges over

the entire admissible parameter set:

$$C = \frac{\int_{\mathcal{A}_a} E\{[\mathbf{x}(t) - \hat{\mathbf{x}}(t)]^T \mathbf{W} [\mathbf{x}(t) - \hat{\mathbf{x}}(t)]\} d\mathbf{a}}{\int_{\mathcal{A}_a} d\mathbf{a}} \quad (60)$$

Note that

$$\int_{\mathcal{A}_a} d\mathbf{a} \triangleq \int_{\mathcal{A}_p} \dots \int_{\mathcal{A}_2} \int_{\mathcal{A}_1} da_1 da_2 \dots da_p$$

where p is the number of scalar parameters (the dimension of \mathbf{a}), and where \mathbf{W} is a weighting matrix chosen by the designer to place emphasis on certain states. Also, define \mathcal{A}_a as the admissible set of parameter values, where each parameter varies over a continuous bounded range. A five-step algorithm was developed by Sheldon which allows the designer to approximate and minimize the cost functional numerically [68, 69]. This procedure is accomplished prior to the real-time implementation of the (fixed-bank or moving-bank) MMAE and provides the designer with the optimal choice for the discretization level. The basic question being addressed is, "If allowed J discretized points (where J is preselected) in the parameter space, where should they be placed in order to yield optimal MMAE performance relative to the chosen cost function C ?" In general, the three cost functions described above will yield different discretizations; so the designer must determine which cost criterion is most pertinent to the problem at hand.

With the focus of this research on parameter estimation, the cost functional for parameter estimation [68, 69],

$$C = \frac{\int_{\mathcal{A}_a} E\{[\mathbf{a}(t) - \hat{\mathbf{a}}(t)]^T \mathbf{W} [\mathbf{a}(t) - \hat{\mathbf{a}}(t)]\} d\mathbf{a}}{\int_{\mathcal{A}_a} d\mathbf{a}} \quad (61)$$

is required. The following summary is taken from Sheldon's dissertation [68] and highlights his assumptions, concepts, and presents his design algorithm. Some assumptions, besides those stated

earlier for the MMAE development, are required to bound the problem and form the basis for the mathematical development:

1. The structure of the plant is known except for an p -vector of parameters, which is assumed to be a member of an infinite set \mathcal{A}_a , which is Lebesgue measurable.
2. The set, \mathcal{A}_a , is bounded and connected so that it makes sense to integrate over the set numerically, in order to find the mean square error
3. There are a finite number of filters available for the estimation.
4. *Pseudo*-probabilities are calculated by Equation (44),

$$p_j(t_i) = \frac{f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i | \mathbf{a}_j, \mathbf{Z}_{i-1}) p_j(t_{i-1})}{\sum_{k=1}^J f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i | \mathbf{a}_k, \mathbf{Z}_{i-1}) p_k(t_{i-1})}$$

for $j = 1, 2, \dots, J$. The probabilities are referred to as pseudo-probabilities since it is impossible to have an infinite bank of filters, hence the MMAE algorithm might be calculating the probability that each filter is the correct one, when in fact none of them are.

5. The Bayesian estimate formulations in Equations (50) and (51), are used to calculate the estimates:

$$\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+) = \sum_{j=1}^J \hat{\mathbf{x}}_j(t_i^+) p_j(t_i)$$

$$\hat{\mathbf{a}}_{\text{MMAE}}(t_i) = \sum_{j=1}^J \mathbf{a}_j p_j(t_i)$$

6. The MMAE system converges to the model closest, in the "Baram sense" (in the information

measure of Baram [4]), to the true parameter value with probability one.

7. The development assumes steady state, constant-gain filters, i.e., the \mathbf{K}_j 's do not vary with time. However, Sheldon points out that the filters in the MMAE do not have to be steady state Kalman filters. But for this development to apply, the filters must have the same structure as a Kalman filter. Therefore, any other method to calculate a constant gain that produces a stable estimator may be used [68,69]. For example, the problem researched in Chapter 4 is an unstable, nonlinear, integrated GPS/INS problem using extended Kalman filters. A nominal trajectory point is chosen as the basis for defining the system matrices for determining a "*pseudo*"-constant gain value. Once the gains are calculated, the Sheldon algorithm may be applied using a finite horizon assumption to generate an approximate solution, since the system never achieves steady state. The finite horizon is chosen by the designer and based on what finite period of time is of physical concern for the problem. Specifically, the finite horizon is given by the number of sample periods into the future that should be used to propagate the iterative relationship given by Equation (68).

Given these assumptions, the goal is to minimize the cost functional

$$C = \frac{\int_{\mathcal{A}_a} E\{[\mathbf{a}(t) - \hat{\mathbf{a}}(t)]^T \mathbf{W}[\mathbf{a}(t) - \hat{\mathbf{a}}(t)]\} d\mathbf{a}}{\int_{\mathcal{A}_a} d\mathbf{a}} \quad (62)$$

Since the admissible parameter set is assumed constant for a given problem, only the numerator of Equation (62),

$$\int_{\mathcal{A}_a} E\{[\mathbf{a}(t) - \hat{\mathbf{a}}(t)]^T \mathbf{W}[\mathbf{a}(t) - \hat{\mathbf{a}}(t)]\} d\mathbf{a} = \int_{\mathcal{A}_a} \text{tr}(\mathbf{W} E\{[\mathbf{a}(t) - \hat{\mathbf{a}}(t)][\mathbf{a}(t) - \hat{\mathbf{a}}(t)]^T\}) d\mathbf{a}$$

needs to be minimized. This simply entails a sensitivity analysis into the effects of parameter variations in the elemental filters of the MMAE.

The error autocorrelation equations are derived from the standard Kalman filter equations (32) and (33) based on the design model hypothesizing the parameter value \mathbf{a}_j ,

$$\mathbf{x}_j(t_i) = \Phi_j(t_i, t_{i-1})\mathbf{x}_j(t_{i-1}) + \mathbf{G}_{dj}(t_{i-1})\mathbf{w}_{dj}(t_{i-1}) \quad (63)$$

$$\mathbf{z}(t_i) = \mathbf{H}_j(t_i)\mathbf{x}_j(t_i) + \mathbf{v}_j(t_i) \quad (64)$$

which varies from the actual truth system based on the true parameter \mathbf{a}_t :

$$\mathbf{x}_t(t_i) = \Phi_t(t_i, t_{i-1})\mathbf{x}_t(t_{i-1}) + \mathbf{G}_{dt}(t_{i-1})\mathbf{w}_{dt}(t_{i-1}) \quad (65)$$

$$\mathbf{z}_t(t_i) = \mathbf{H}_t(t_i)\mathbf{x}_t(t_i) + \mathbf{v}_t(t_i) \quad (66)$$

Given these standard equations, the estimation error vector is given by

$$\bar{\mathbf{x}}_j(t_i^-) = \hat{\mathbf{x}}_j(t_i^-) - \mathbf{T}\mathbf{x}_t(t_i^-) \quad (67)$$

where \mathbf{T} denotes a transformation matrix from the true state space to the model state space, which allows reduced order filter designs.

The appropriate MMAE filter selections are made based on [4] using the one-step prediction model of the state estimate $\hat{\mathbf{x}}_j(t_i^-)$. Sheldon derived the one-step prediction model from the standard Kalman filter equations in terms of the estimation error vector $\bar{\mathbf{x}}_j(t_i^-)$ to produce the iterative error autocorrelation matrix equation (the time arguments, (t_i) , are dropped for convenience) [68, 69]:

$$E \left\{ \begin{bmatrix} \bar{\mathbf{x}}_j^- \\ \mathbf{x}_t^- \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_j^{-T} & \mathbf{x}_t^{-T} \end{bmatrix} \right\} = \mathbf{\Gamma}_j(t_i) = \mathcal{L}\mathbf{\Gamma}_j(t_{i-1})\mathcal{L}^T + \mathbf{G}_0\mathbf{Q}_0\mathbf{G}_0^T \quad (68)$$

where

$$\mathcal{L} = \begin{bmatrix} \Phi_j(\mathbf{I} - \mathbf{K}_j\mathbf{H}_j) & \mathbf{T}\Delta\Phi - \Phi_j\mathbf{K}_j\Delta\mathbf{H} \\ \mathbf{0} & \Phi_t \end{bmatrix} \quad (69)$$

$$\mathbf{G}_0 = \begin{bmatrix} -\mathbf{T}\mathbf{G}_{dt} & \Phi_j \mathbf{K}_j \\ \mathbf{G}_{dt} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{Q}_0 = \begin{bmatrix} \mathbf{Q}_{dt} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_t \end{bmatrix}$$

and

$$\mathbf{T}\Delta\Phi \triangleq \Phi_j \mathbf{T} - \mathbf{T}\Phi_t$$

$$\Delta\mathbf{H} \triangleq \mathbf{H}_j \mathbf{T} - \mathbf{H}_t$$

Notice that \mathcal{L} , \mathbf{G}_0 , and \mathbf{Q}_0 are formed by partitioned matrices. If \mathcal{L} is a contraction, as $t_i \rightarrow \infty$, $\Gamma_j(t_i)$ approaches a constant value which is the steady state prediction error autocorrelation, and is denoted Γ_j^∞ . The upper left partition of $\Gamma_j(t_i)$, $E \left\{ \begin{bmatrix} \bar{\mathbf{x}}_j^- \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_j^- \end{bmatrix}^T \right\}$, is Ψ_j^- , the autocorrelation of the estimator prediction error for the j^{th} filter, and $E \left\{ \begin{bmatrix} \bar{\mathbf{x}}_j^- \end{bmatrix} \mathbf{W} \begin{bmatrix} \bar{\mathbf{x}}_j^- \end{bmatrix}^T \right\}$ is calculated by:

$$E \left\{ \begin{bmatrix} \bar{\mathbf{x}}_j^- \end{bmatrix} \mathbf{W} \begin{bmatrix} \bar{\mathbf{x}}_j^- \end{bmatrix}^T \right\} = tr \left[\mathbf{W} \Psi_j^- \right] \quad (70)$$

This value can be calculated at any point in the parameter space once the filter to which the MMAE converges is identified. Given that there are sufficient conditions for the MMAE to converge in "Baram sense" [4], it will converge to the j^{th} filter governed by [68, 69]:

$$\ell_j = \min \ell_k \quad k = 1, \dots, J \quad (71)$$

where ℓ_k is defined as the proximity of the k^{th} filter generated by [4] as:

$$\ell_k \triangleq \log_e |\mathbf{A}_k| + tr \left\{ [\mathbf{A}_k]^{-1} \left(\begin{bmatrix} \mathbf{H}_k & \Delta\mathbf{H} \end{bmatrix} \Gamma_k^\infty \begin{bmatrix} \mathbf{H}_k & \Delta\mathbf{H} \end{bmatrix}^T + \mathbf{R}_t \right) \right\} \quad (72)$$

given

$$\mathbf{A}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \quad (73)$$

Careful attention should be paid to Equation (72) recognizing that the notation $\begin{bmatrix} \mathbf{H}_k & \Delta\mathbf{H} \end{bmatrix}$ indicates partitioned matrices.

Sheldon developed a five-step algorithm allowing the designer to approximate and minimize the cost functional given in Equation (60) or (62) numerically :

1. Start by describing the system in terms of the parameter vector \mathbf{a} , specifying the structure of the truth system and the filters to be implemented in the estimator.
2. Choose J , the number of filters to be implemented in the estimator.
3. Choose a representative parameter set to begin the minimization. The set should be chosen based on which cost functional is being minimized:
 - For the cost function that penalizes *state estimator* errors, a reasonable choice can be obtained by plotting the optimal estimation error autocorrelation, Equation (70), for a coarse discretization of the parameter space assuming $\mathbf{a}_j = \mathbf{a}_i$ at each point. The larger $\text{tr} [\mathbf{W}\Psi_i^-]$ is, the closer the representative parameters should be.
 - For the cost function that penalizes *parameter estimator* errors, a reasonable choice may be made by equally dividing the admissible parameter region into equal intervals and choosing the midpoint of each interval as a member of the representative parameter set.
4. Use a numerical integration technique to evaluate Equation (60) or (62) for any given choice of a representative parameter set. The functional evaluations required at each interval are calculated with Equation (68). The information required to set up Equation (68) is based on the proper filter selection per Equation (72). Note, for the cost functional pertaining to the parameter estimator, the value of $\hat{\mathbf{a}}$ needed in Equation (62) is the value of \mathbf{a} forming the basis of the filter selected at the evaluation point.
5. Step 4 generates a numerical approximation to the value of $C(\mathbf{a})$ for a prespecified choice of parameter vector \mathbf{a} . The problem becomes a vector minimization problem, and a vector minimization technique can be used to minimize $C(\mathbf{a})$ using the procedure in step 4 to evaluate the functional for each parameter vector.

To avoid estimator lockup, Sheldon extended his work to account for the design practice of placing lower bounds on the elemental filter probabilities, p_j . The lower bounds, p_{\min} , are used to ensure that the system is adaptable to true parameter value changes [68,69]. The parameter estimate is then given by

$$\begin{aligned}\hat{\mathbf{a}}(t_i) &= \sum_{j=1}^J \mathbf{a}_j p_j(t_i) \\ &= \left\{ [1 - (J \cdot p_{\min})] \mathbf{a}^{sel}(t_i) \right\} + \sum_{j=1}^J \mathbf{a}_j(t_i^+) p_{\min}\end{aligned}\quad (74)$$

where \mathbf{a}^{sel} indicates the filter-assumed parameter value selected via Equation (71), i.e., the j^{th} filter to which the MMAE converged in the “Baram sense” [4]. The cost function in Equation (62) is now solved using $\hat{\mathbf{a}}(t_i)$ from Equation (74).

Sheldon’s algorithm will be modified to provide on-line discretization of an adaptive MMAE bank, including the use of a finite horizon discussed previously. Specifically, a moving-bank MMAE will be incorporated with the modified on-line Sheldon discretization, as presented in Section 3.1.5.4.

2.2.4 Hierarchical Structure

Maybeck and his students have researched multiple failures for reconfigurable flight control via MMAE methods [11, 14, 15, 50–52] and the following section parallels the discussion presented by Miller [53]. If a multiple model algorithm were based upon all possible single and double failures of K sensors, it would require one elemental filter for the fully functional status, K single-failure elemental filters, and $K!/(K-2)!2!$ double-failure filters. To avoid this computational burden, the idea of the “moving-bank” leads to the concept of a hierarchical structure that requires at most only $(J=K+1)$ elemental filters to be on-line at any given time: the same number used when only single failures are modeled. Figure 11 illustrates such a hierarchical structure. At “Level Zero”, there

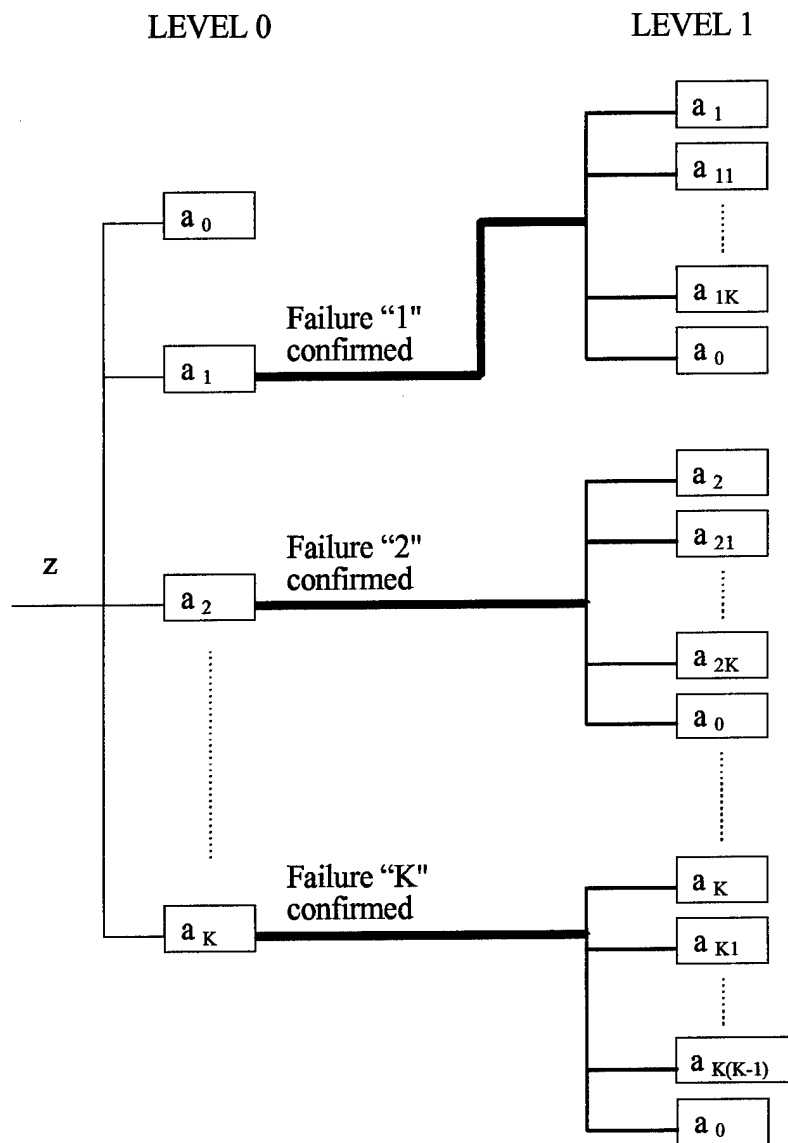


Figure 11. Hierarchical Modeling - Level 0 and Level 1 MMAE Banks

are K elemental filters specifically designed for one of the single-failure conditions and one configured for the fully functional system (denoted as \mathbf{a}_0 in the figure). Upon confirmation that failure “ k ” (any of the possibilities $1, 2, \dots, K$) has occurred, a new MMAE bank is brought on-line from memory at “Level One” and replaces the original “Level Zero” MMAE. It would consist of $J=K+1$ elemental filters: one designed for the k^{th} single-failure condition (denoted \mathbf{a}_k), $K-1$ configured for the double-failure condition of the known k^{th} failure plus one of the remaining possible failures (denoted $\mathbf{a}_{k,l}$), and one designed for the fully functional (\mathbf{a}_0) system to allow for using future measurements to change the decision that the first failure, had, in fact, occurred.

2.2.5 Inter-Residual Distance Feedback (IRDF)

The following section parallels the discussion presented by Miller [53]. The common assumption used when employing MMAE techniques, is that the parameters take on only a finite number of different values. An MMAE’s successful operation depends on the distinguishability of the models used and the tuning of the filters based on the parameters chosen. There must be significant differences in the characteristics of the residual in the “correct” versus “mismatched” filters. Each filter should be tuned for best performance when the “true” values of the unknown parameters are identical to its assumed value for these parameters. When Kalman filters are used in the bank, conservative tuning should be avoided to prevent the residuals from becoming too *close* together and affecting the discrimination property of the filter bank. For fast and reliable discrimination, the residuals should be as distinct as possible [36, 45].

Lund has proposed a modification to the MMAE concept and has successfully demonstrated it, by simulation, for a second order single-input single-output system [35, 36]. The method, Inter-Residual Distance Feedback (IRDF), provides for on-line modification of the elemental filters for the purpose of maintaining the discrimination property of the MMAE filter bank. The method

modifies the elemental filters to keep the predicted measurements from becoming too close in “some sense”, thus affecting the distinguishability of the elemental filters and thereby the properties of the MMAE algorithm. The elemental filters are modified by detuning the filters through modulation of either the dynamic driving noise \mathbf{Q}_{dj} or the new information $\mathbf{K}_j \mathbf{r}_j$ directly. Recall that \mathbf{K}_j is the j^{th} elemental filter gain matrix and \mathbf{r}_j is the residual vector of the j^{th} elemental filter. Modulation is governed by a scalar quantity computed from a distance measure between the residuals. The IRDF concept was initially applied to continuous-time systems by Lund [35, 36] and extended to the discrete-time case by Miller [53].

2.2.6 Model-Group Switching

A new algorithm has been developed that includes the additional flexibility of changing the number of filters in the MMAE bank [31]. Notice that all the other methods discussed to this point require the number of filters in the bank to remain fixed. The new algorithm is called model-group switching (MGS) since it relies on groups of models which are closely related in terms of system behavior and switches between these model groups. Switching refers to choosing which model groups are included in the MMAE at each sample time. More than one model group can be active at a given time and later deactivated based on the decision logic. Clearly the number of filters in the bank as well as the span of the bank in parameter space would change as the switching logic activates and deactivates model groups. Consider a two-dimensional parameter estimation problem. Several model groups could be dedicated to each parameter. If a parameter variation is detected in one dimension, then the model groups associated with that parameter would be candidates for inclusion into the bank. Once the parameter estimate adapts and is deemed adequate, then either the original model group or the newly activated model group could be deactivated to reduce computational loading.

2.3 Chapter Summary

This chapter presented the background theories that lay the foundation for the new concepts developed in Chapter 3. Some of the items discussed here, such as chi-square tests and SPRTs, provided insights into the development of new hypothesis testing ideas and analytical means of selecting decision thresholds. Sheldon's discretization technique and the moving-bank MMAE will receive considerable attention in the next three chapters. All of the ideas presented in this chapter, including the basic voting method which led to combining decision making measures with logical AND's and OR's, played an important role in the development that follows.

Chapter 3 - Theory Development

This chapter presents the theory developed in support of the dissertation research. Section 2.2.3 introduced the fundamental concepts of a moving-bank MMAE. The focus of the research has been to develop algorithms designed to direct the placement of the parameter points to be used as the basis of the elemental filters in a moving-bank MMAE. It is desired to exploit the available information as fully as possible, to make decisions to move, expand and contract the bank in a better and more systematic fashion than is currently possible with the *ad hoc* techniques described in Section 2.2.3. To this end, two new algorithms have been formulated and are titled the “Density Algorithm” and the “Probability Algorithm”. The density algorithm is subject to many *ad hoc* methods including empirically determined thresholds, but some progress has been made in the development of analytically determined thresholds. Additionally, this algorithm presents the first steps in the exploitation of density data available from the MMAE as described in the next section. Finally, research into portions of the density algorithm led to the development of the probability algorithm, which is much more analytical in both its derivation and implementation. The sections in this chapter describe the thought processes behind the development of the algorithms, including the underlying theories and the details of the final implementations.

Figure 12 shows the major topics of interest related to the moving-bank MMAE as (1) deciding when and where to move the bank (2) deciding when and how to expand or contract the bank (3) determining the level of discretization for the bank and (4) other techniques such as model-group switching and IRDF. The newly developed algorithms address major topics (1) - (3), and comparisons to previous methods are discussed theoretically in this chapter and through simulation results in Chapter 4.

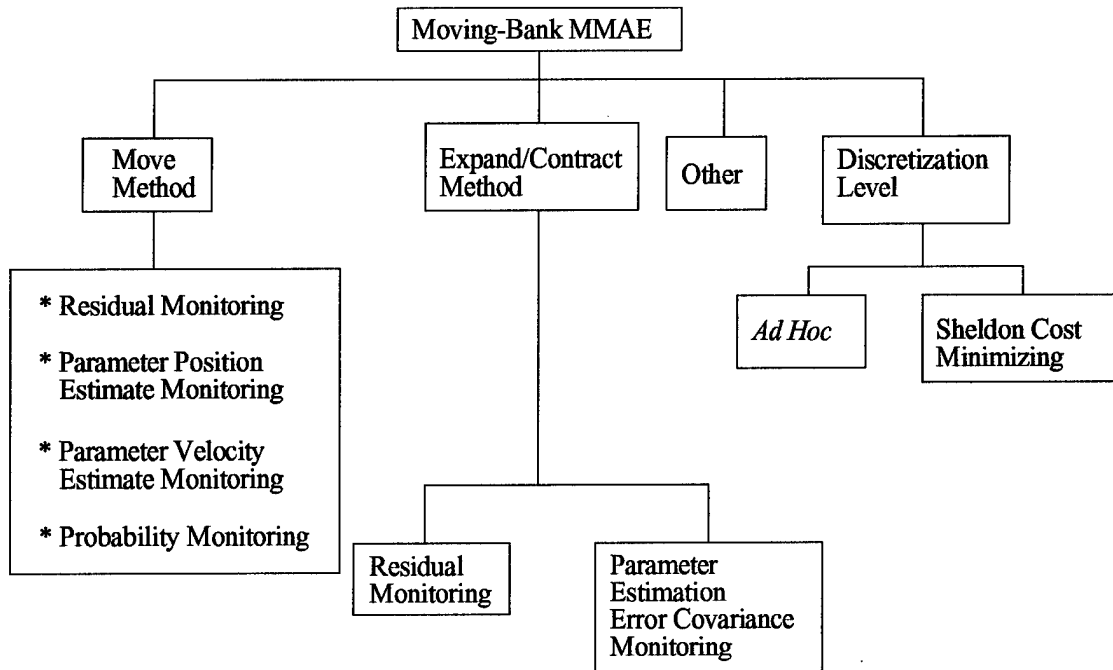


Figure 12. Moving-Bank MMAE Topics of Interest

3.1 Density Algorithm

Since considerable research had already been conducted in the realm of moving-bank MMAE's, the primary question being asked at the onset of this research was "what new information can be exploited to derive a new moving-bank algorithm?" The answer is contained in the conditional density of the current measurements, conditioned on the assumed value of the parameter vector and the observed values of the previous measurement history, $f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathbf{Z}_{i-1})$, which will be referred to as the density $f_{\mathbf{z}_i|\mathbf{a}, \mathbf{Z}_{i-1}}$ for simplicity. Recall that each elemental filter in the MMAE bank provides a sample of this density function, each for a different value of \mathbf{a}_j . The sample values are

calculated using Equation (47), which is repeated here. The density samples are given by

$$f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathbf{Z}_{i-1}) = \beta_j \exp\{\cdot\} \quad (75)$$

where

$$\beta_j = \frac{1}{(2\pi)^{m/2} |\mathbf{A}_j(t_i)|^{1/2}} \quad (76)$$

and the expression in the brackets is

$$\{\cdot\} = \left\{ -\frac{1}{2} \mathbf{r}_j^T(t_i) \mathbf{A}_j^{-1}(t_i) \mathbf{r}_j(t_i) \right\} \quad (77)$$

The following subsections discuss the details of a new algorithm based on the above density, beginning with its development through data analysis, followed by a description of the logic flow, and ending with enhancements that improved the algorithm's performance.

3.1.1 Algorithm Development

Although this algorithm is focussed on the density $f_{\mathbf{z}_i|\mathbf{a}, \mathbf{Z}_{i-1}}$, the actual density that is more motivating to monitor and characterize is $f_{\mathbf{a}|\mathbf{Z}_i}$. Note the subtlety in the subscripts which identify this function as the conditional density of the parameters based on the measurement history. The underlying concept behind utilizing the density $f_{\mathbf{z}_i|\mathbf{a}, \mathbf{Z}_{i-1}}$ is to determine the shape of the conditional density $f_{\mathbf{a}|\mathbf{Z}_i}$. A natural question would be, why not consider $f_{\mathbf{a}|\mathbf{Z}_i}$ itself directly, and this will be answered as the discussion develops. The next step might be to center around or contract about the highest peak of $f_{\mathbf{a}|\mathbf{Z}_i}$ in the parameter space, as this will appropriately yield the placement of the elemental filters about the region of highest probability. This concept is motivated by the relationship between the two density functions which is shown here and derived by Maybeck [45]:

$$f_{\mathbf{a}|\mathbf{Z}_i} = \frac{f_{\mathbf{z}_i|\mathbf{a}, \mathbf{Z}_{i-1}} f_{\mathbf{a}|\mathbf{Z}_{i-1}}}{\int_{\mathcal{A}_{\mathbf{a}}} f_{\mathbf{z}_i|\mathbf{a}, \mathbf{Z}_{i-1}} f_{\mathbf{a}|\mathbf{Z}_{i-1}} d\alpha} \quad (78)$$

Note that \mathbf{a} can assume any value in a continuum set $\mathcal{A}_{\mathbf{a}} \subset \mathbb{R}^p$ and α is a dummy variable for the parameter vector. One could simply distribute the discrete parameter values to be used for the basis of the elemental filters, uniformly about the peak density value. Alternatively, nonuniform

distributions could be used based on the shape of the density. The problem is that the shape of $f_{\mathbf{a}|\mathbf{Z}_i}$ or its peak value are not known but merely J discretized values of it are available, where J is the number of elemental filters in the bank. If the parameters could only assume one of J discrete values given by $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_J$, then Equation (??) would yield

$$f_{\mathbf{a}|\mathbf{Z}_i} = \sum_{j=1}^J p_j(t_i) \delta(\alpha - \mathbf{a}_j) \quad (79)$$

where $p_j(t_i)$ is given by Equation (44), repeated here:

$$p_j(t_i) = \frac{f_{\mathbf{Z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{Z}_i|\mathbf{a}_j, \mathbf{Z}_{i-1}) p_j(t_{i-1})}{\sum_{k=1}^J f_{\mathbf{Z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{Z}_i|\mathbf{a}_k, \mathbf{Z}_{i-1}) p_k(t_{i-1})} \quad (80)$$

However, we actually wish to conceive of $f_{\mathbf{a}|\mathbf{Z}_i}$ as a continuous surface for which Equation (79) is to be viewed as a discretized approximation (the $p_j(t_i)$ are then “volumes” under that surface over areas surrounding each \mathbf{a}_j) that will change with the number and location of the discrete values \mathbf{a}_j . However, the approximating density given by Equation (79) is normalized in the above equation such that $\sum_{j=1}^J p_j = 1$; so the probabilities only give an *indication* of the density shape *within* the region covered by the bank. In other words, it assumes the density function is essentially zero outside the bank and thus distorts the true shape of the density at the sampled values. For instance, if the bank were currently located far from the true parameter, then *all* of the likelihood quotients given by Equation (54) would be very large (indicative of the bank placement problem), and thus *all* of the $f_{\mathbf{Z}_i|\mathbf{a}, \mathbf{Z}_{i-1}}$ values given by Equation (75) would be correspondingly small, but the $p_j(t_i)$ ’s computed by Equation (80) would always add to one and *not* reveal the problem. This motivates exploiting the unnormalized samples of $f_{\mathbf{Z}_i|\mathbf{a}, \mathbf{Z}_{i-1}}$ with a newly developed decision logic. Additionally, note

that $p_j(t_i)$ in Equation (80) is a function of $f_{\mathbf{z}(t_i)|\mathbf{a},\mathbf{Z}(t_{i-1})}$ and $p_j(t_{i-1})$ (the elemental probability value at the *previous* sample time), so the density, $f_{\mathbf{z}(t_i)|\mathbf{a},\mathbf{Z}(t_{i-1})}$, provides *lead*-type information about the probabilities and anticipates changes in these probabilities. This implies that a decision logic based on $f_{\mathbf{z}(t_i)|\mathbf{a},\mathbf{Z}(t_{i-1})}$ has the potential to react to changes in the true parameter value more quickly than a decision logic based on $p_j(t_i)$. This implication is supported through simulation results presented in Chapter 4.

Now that the unnormalized samples of $f_{\mathbf{z}_i|\mathbf{a},\mathbf{Z}_{i-1}}$ were identified as the “quantities of interest”, extensive empirical analysis was conducted to determine useful trends in this piece of data. Computer simulations were performed using a filter performance evaluation tool named MMSOFE [60] that subjects a proposed MMAE design against a “truth model” simulation of the real world (and then conducts a Monte Carlo analysis), and a 13-state GPS / INS navigation model detailed in Chapter 4. In this particular application, the uncertain parameter is a scalar, the covariance R_{GPS} of the noise in the GPS measurements, given by Equation (35) and appearing in Equations (37) – (51) and explicitly in Equation (39). The empirical analysis began by simply observing the raw density data plotted in MATLAB [41] and manually identifying trends that appeared in the elemental filter densities when parameter variations were induced into the truth model. The manual analysis led to investigations into what types of *measures* of the raw data would provide an algorithm with indicators useful in a decision logic. Candidate measures were analyzed based on consistency of making the desired (appropriate) decisions such as move, expand or contract the bank in the presence of parameter changes. Once the candidate measures were classified as accepted or rejected, a preliminary decision-making process was conceived and implemented. A short study was performed to recognize both desirable and undesirable trends in the preliminary algorithm. Additional measures and techniques were analyzed, leading to the final algorithm.

3.1.2 Candidate Measures Rejected

The following paragraphs describe the measures that were considered for use in the density algorithm but rejected. Reasons for rejecting these relative to other possible measures are discussed for each measure, and then attention will be focussed on the accepted measures in Section 3.1.3. Candidate measures that were rejected are labeled m_i whereas those that were accepted are labeled M_i .

Sum of density values: This measure was intended to indicate whether the filters assuming the current parameter values were placed in a region of high or low probability in terms of the true density $f_{\mathbf{z}_i|\mathbf{a},\mathbf{Z}_{i-1}}$. This is best illustrated in simplified drawings of the density function for a one-dimensional problem. Figure 13 shows that a good choice of parameter values $a_1 - a_3$ (i.e., close to the true parameter value) would result in all large values for the density function samples. Note carefully that Figure 13 is a plot of $f_{\mathbf{z}_i|\mathbf{a},\mathbf{Z}_{i-1}}(\zeta_i|\alpha,\mathbf{Z}_{i-1})$ versus the parameter α rather than versus the incoming measurement ζ_i , as might have been anticipated. In the context of an MMAE, at time t_i we know ζ_i and \mathbf{Z}_{i-1} but we do not know the best value (or values) of α , so this makes sense.

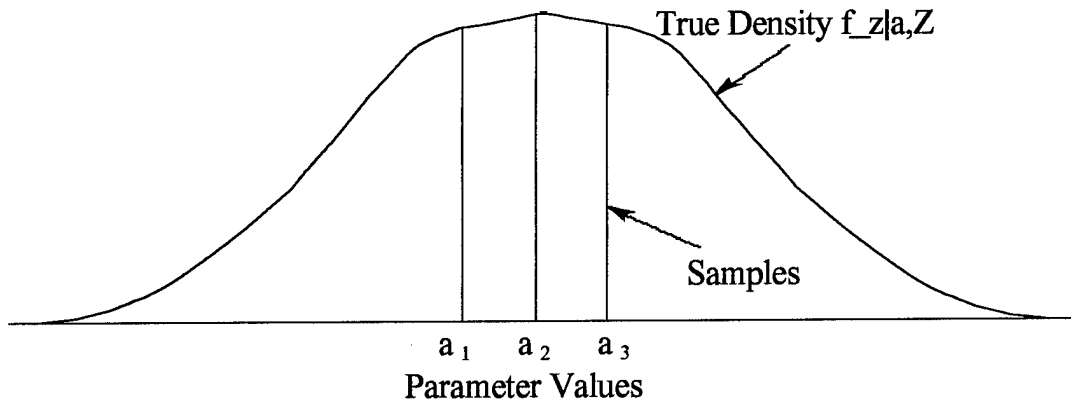


Figure 13. All Density Values Relatively Large

If ζ_i and α were both scalar-valued, then one could conceive of plotting $f_{\mathbf{z}_i|\mathbf{a},\mathbf{Z}_{i-1}}(\zeta_i|\alpha,\mathbf{Z}_{i-1})$ as a surface over the $\zeta_i - \alpha$ plane (let ζ_i rise vertically out of the page in Figure 13). Then one could slice through that surface with a plane at the known (observed) value of ζ_i , which would yield the plane of the previous page and Figure 13 as the appropriate slice out of the surface.

Similarly, Figure 14 shows that a poor choice of parameter values $a_1 - a_3$ (i.e., far from the true parameter value) would result in all small values for the density function samples. The resulting measure

$$m_1 = \sum_{j=1}^J f_{\mathbf{z}|\mathbf{a},\mathbf{Z}}(j)$$

could then be used as a basis for decision making. Unfortunately, this measure gave inconsistent trends, as its order of magnitude changed significantly as a function of the true parameter value rather than how well the filter-assumed parameter values matched the true parameter value (recall that the uncertain parameter is the measurement noise covariance, $\mathbf{R}(t_i)$). For example, if the true parameter value changed by an order of magnitude, then the measure would follow suit regardless of whether or not the filter assumed-values were well placed with respect to truth.

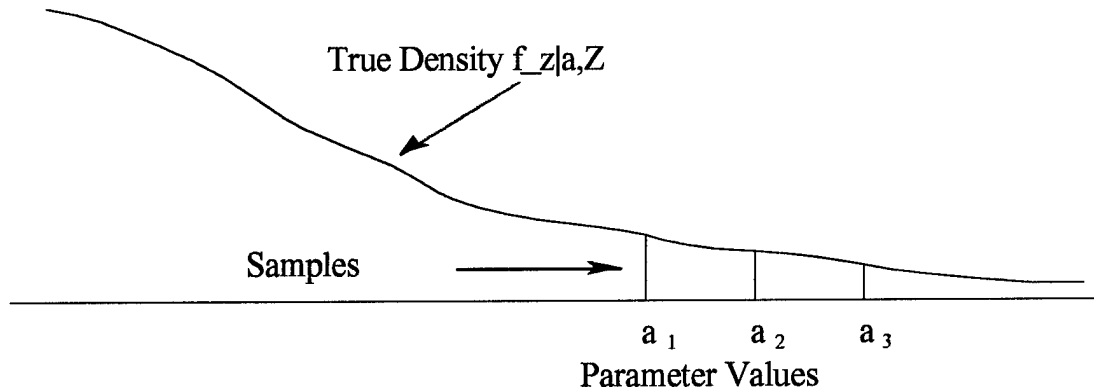


Figure 14. All Density Values Relatively Small

Sum of delta density values: The phrase *delta density value* refers to the difference between the density samples of two adjacent elemental filters within the bank. The filters are placed within the bank such that their assumed parameter values are in ascending order. The sum of the absolute value of these delta values is used to form rejected measure m_2 :

$$m_2 = \sum_{j=1}^{J-1} |f_{\mathbf{z}|\mathbf{a},\mathbf{z}}(j) - f_{\mathbf{z}|\mathbf{a},\mathbf{z}}(j+1)|$$

Notice that the absolute value of the delta density is used to gain insight into the variations of the density samples. A lack of variation of the density samples would indicate that the filters are located close to truth, provided that m_1 is simultaneously large, and thus preclude the algorithm from making unnecessary bank movements. Figure 13 shows such a case where the density variations would be small. A lack of variation while m_1 is simultaneously very small could indicate being very far from the true parameter value, as seen in Figure 14. In contrast, Figure 15 shows the case where large variations and thus a large value for m_2 would exist, leading to the desire to move the filter bank closer to the density peak, i.e., to the right. This measure also gave inconsistent trends as its order of magnitude changed significantly as a function of the true parameter value. This was later rectified by normalizing the delta density values, resulting in the accepted measure M_5 discussed presently.

Curvature and slope of spline fit [m_{3c} and m_{3s}]: Recall that J samples of the density function are available, where J is the number of filters in the bank. The intent is to gain insight into the shape of the density function based solely on these J samples and make a determination where the elemental-filter-assumed parameter points lie in the parameter space with respect to truth. The

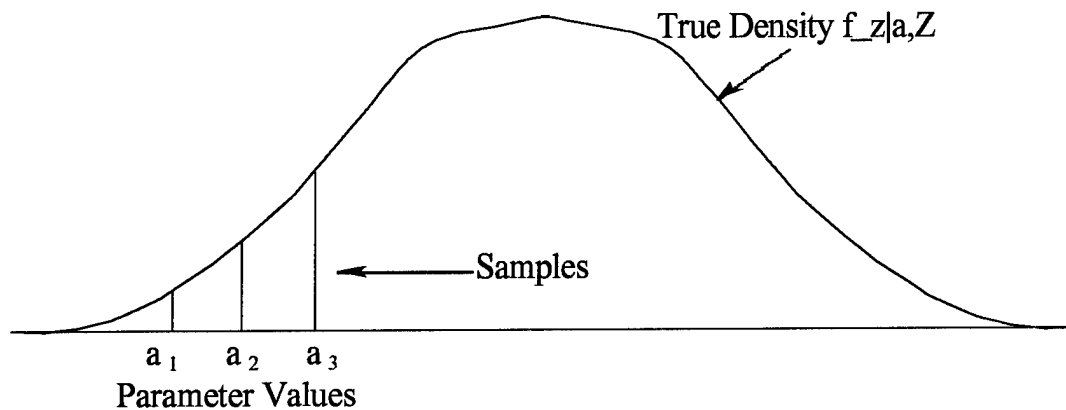


Figure 15. Large Variations in Density Values

density samples for each filter are plotted at a given sample time and a spline fit is performed. The measures, m_{3s} and m_{3c} , represent the first and second derivatives of the spline fit evaluated at specified points. MATLAB provides toolbox functions for performing both the spline fit and calculation of its derivatives [41]. The curvature, m_{3c} , or second derivative of the spline fit is evaluated at the maximum density sample and is used to help determine the appropriate span of the filter-assumed parameter values within the bank. For instance, a *large* negative curvature indicates that the density function has a *narrow* peak and suggests using a *small* bank span, while a *small* negative curvature which indicates that the density function has a *broad* peak and suggests using a *large* bank span. Additionally, m_{3c} could provide information about the location of the *true* density peak with respect to the current density samples. For example, a positive curvature would result from the density samples shown in Figures 14 or 15, giving an indication that the *true* density peak is to the left or right, respectively. A negative curvature would result from the density samples shown in Figure 13, giving an indication that the *true* density peak is within the range spanned by the filter-assumed parameter values. Additional insights could be obtained by observing the sign of

the slope, m_{3s} , or first derivative of the spline fit evaluated at each density sample, since a change from positive to negative slope would indicate that the true density peak lies between the elemental density samples associated with this sign change.

The spline fit proved to be computationally intensive and suffered oscillations about the zero crossing when the filter-assumed parameter values spanned a large range, as shown in Figure 16. Oscillations about the zero crossing is a natural occurrence when attempting to curve fit data that are separated in the abscissa by orders of magnitude. Nevertheless, this made the measure based on the slope and curvature of the spline fit unreliable for consistent decision making, since density functions clearly cannot be negative. Constrained best fits, using nonnegative splines, and other means of rectifying this problem were considered. However, this was eventually abandoned because some of the other measures (in Section 3.1.3) provided consistently useful information that was analogous to this measure's information, but with much less computational load.

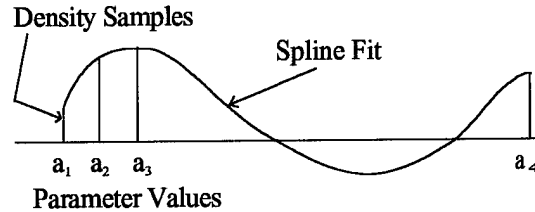


Figure 16. Spline Fit of Density Samples

Measures of the likelihood quotient and residual covariance: Likelihood quotient monitoring was implemented to help identify the “correctness” of a given filter. Thus $m_4 = L_j(t_i)$ for $j = 1, 2, \dots, J$. Likelihood quotient monitoring utilizes the likelihood quotient, $L_j = \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$, which is compared to a threshold value. If the threshold is exceeded, the filter is considered a bad match to truth, i.e., it has relatively large residuals. However, if L_j is below the threshold, then the filter

is considered a good match to truth, i.e., it has relatively small residuals. This information was intended to provide insights into how harsh a movement should be, i.e., hard, firm or soft. These relative terms for the harshness of a move indicate how far the bank is moved in the parameter space and will be explained in Section 3.1.4.2. The upcoming discussion of measure M_7 details the successful use of likelihood quotient monitoring to identify poorly matched filters.

It was also proposed that the likelihood quotient could provide additional information about which direction to move the bank or when to expand/contract the bank. Preliminary analysis indicated a need to rescale L_j by some measure of the matrix \mathbf{A}^{-1} prior to comparing it to the threshold. The reason is that the quadratic $\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$ was dominated by large values of the filter-assumed parameter (i.e., large entries in the \mathbf{R}_j matrix), regardless of changes in the residuals, \mathbf{r}_j . Recall that large residual values relative to the filter-computed residual covariance $\mathbf{A}_j(t_i)$ (and thus a large likelihood quotient) indicate that the filter is a poor match to truth, whereas small residual values relative to $\mathbf{A}_j(t_i)$ (and thus a small likelihood quotient) indicate the filter is a good match to truth. However, the filter-assumed measurement noise covariance, \mathbf{R}_j , was varied by three orders of magnitude between elemental filters, and the filter-computed residual covariance matrix was calculated using $\mathbf{A}_j = \mathbf{H}_j \mathbf{P}_j^- \mathbf{H}_j^T + \mathbf{R}_j$. Therefore, the likelihood quotient was dominated by the term \mathbf{A}_j^{-1} and masked the useful information contained in the residuals. An attempt was made to rescale L_j using measures $m_5 - m_7$ shown here:

$$m_5 = L_j \text{ scaled with } \text{Trace}(\mathbf{A}^{-1})$$

$$m_6 = L_j \text{ scaled with } \text{Det}(\mathbf{A}^{-1})$$

$$m_7 = L_j \text{ scaled with sum of all elements of } \mathbf{A}^{-1}$$

All three measures failed to provide a consistent method for rescaling L_j . An alternate logic was developed that focused on the similarity of the L_j values for each filter and is presented in the discussion of measures M_8 and M_9 .

3.1.3 Candidate Measures Accepted

The following paragraphs describe the measures that were considered for use in the density algorithm and accepted. Recall that candidate measures that were rejected are labeled m_i whereas those that were accepted are labeled M_i .

Maximum Density Value: This measure simply takes on the maximum density sample value from within the filter bank at each sample time.

$$M_1 = \max_j [f_{\mathbf{z}|\mathbf{a},\mathbf{z}}(j)] \quad j = 1 \dots J \quad (81)$$

The primary use of this measure is to normalize the density samples by dividing M_1 into some function of the elemental densities (see measures M_5 and M_6). The need to normalize the measures became apparent when analyzing the raw data. Since the measures are often compared to threshold values, it is necessary that these thresholds be applicable in the presence of the true parameter variations. However, in many simulation cases in which the true parameters were required to vary significantly, the density samples and thus the measures were badly scaled, disguising any useful information. The *relative* magnitudes of the density samples and measures did present the type of information needed to make good moving-bank decisions; so the density samples were simply rescaled by the maximum value, M_1 , and consistent decision making ensued.

Filter Index Associated with the Maximum Density Value: With the filters numbered from 1 to J , the measure $M_2 \in [1, J]$ is the integer value associated with measure M_1 :

$$M_2 = \arg \left\{ \max_j [f_{\mathbf{z}|\mathbf{a},\mathbf{z}}(j)] \right\} \quad j = 1 \dots J \quad (82)$$

For example, if the maximum density sample is associated with filter 3, then $M_2 = 3$. This measure indicates which direction, left or right, to move the bank, as explained by the following example. Consider a bank of three filters ($J = 3$) in which all three filters are located in parameter space to the left of the *true* peak density value (see Figure 15 on page 63). Since filter 3 is associated with the maximum density sample from within the filter bank, then $M_2 = 3$. The ensuing logic is

$$\text{IF } M_2 = J \Rightarrow \text{Move Right} \quad \text{OR} \quad \text{IF } M_2 = 1 \Rightarrow \text{Move Left}$$

Further insights into this logic are gained by looking at the expression for the conditional probabilities:

$$p_j(t_i) = \frac{f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathbf{Z}_{i-1})p_j(t_{i-1})}{\sum_{k=1}^J f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_k, \mathbf{Z}_{i-1})p_k(t_{i-1})} \quad (83)$$

Measure M_2 is identifying which filter has the largest numerator density term in Equation (83) and thus the largest premultiplier for this iterative probability calculation. This further indicates which filter is attempting to “absorb” more of the probability and become the filter the MMAE thinks is the best match to truth. In this manner, focusing on $f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{Z}(t_{i-1})}(\mathbf{z}_i|\mathbf{a}_j, \mathbf{Z}_{i-1})$ provides a prediction of future values of $p_j(t_i)$, i.e., it provides useful “lead” information about probability flows. The same move logic described previously for measure M_2 can now be applied. If the filter on the right end of the bank (filter J) is the best match to truth, then move the bank to the right. Similar logic is used for moves to the left. The bank would continue to move (left or right) until M_2 becomes associated with a filter other than the ones on the ends of the bank, i.e., $1 < M_2 < J$, and preferably with one near the center of the bank.

Maximum Probability Value: This measure simply takes on the maximum probability value from within the filter bank at each sample time. This measure is not explicitly used but is included

to lend clarification to measure M_4 .

$$M_3 = \max_j [p_j] \quad j = 1 \dots J \quad (84)$$

Filter Index Associated with the Maximum Probability Value: With the filters numbered from 1 to J, the measure $M_4 \in [1, J]$ is the integer value associated with measure M_3 :

$$M_4 = \arg \left\{ \max_j [p_j] \right\} \quad j = 1 \dots J \quad (85)$$

For example, if the maximum probability value is associated with filter 2, then $M_4 = 2$. This measure indicates the filter which best matches the truth model at each sample time. It is used in conjunction with measures M_8 and M_9 to normalize the likelihood quotients used in these measures.

Sum of Normalized Delta Density Values: The phrase *delta density value* refers to the difference between the density samples of two adjacent elemental filters within the bank. The filters are placed within the bank such that their assumed parameter values are in ascending order. The normalized sum of the absolute values of these delta values is used to form measure M_5 :

$$M_5 = \sum_{j=1}^{J-1} \left| \frac{f_{z|a,z}(j) - f_{z|a,z}(j+1)}{M_1} \right| \quad (86)$$

Notice that the absolute value of the delta density is used to gain insight into the variations of the density samples. A lack of variation of the density samples would indicate that the filters are located either close to or far from truth.

First, consider the case shown in Figure 13 on page 60 in which the density variations would be small, and it would be desirable to preclude the algorithm from making unnecessary bank movements. In contrast, Figure 15 on page 63 shows the case in which large variations and thus a large value for M_5 would exist, leading to the desire to move the filter bank closer to the density peak, i.e., to the right. Measure M_5 does not indicate which direction to move the bank but simply that a

move is needed. This dictates the need to combine measures in the decision logic, as described by the flowchart in Section 3.1.4.

Next, consider the case shown in Figure 14 on page 61 in which the density variations would be small (and also their total values would *tend* be small, unlike above), precluding a move by the logic of the preceding paragraph, yet a move would be desirable. This combination of conditions is only likely to occur if the filters are too finely discretized, and the upcoming discussion of measure M_6 will introduce the use of expansions for this scenario. In other words, the lack of distinguishability between the filters is caused by the discretization being too fine, resulting in a small value for M_5 and prevention of a bank movement. Clearly a large bank movement would be better than an expansion, so it will be necessary to check first for movement of the bank and give this precedence over an expansion. This is illustrated in the flowchart presented in Section 3.1.4. In lieu of the bank movement, a bank expansion, or possibly several expansions, would improve the distinguishability between the filters, leading to an increase in M_5 such that a bank movement would be permitted. An alternative and much simpler logic was first considered which relied on the *magnitude* of the density values to identify when the density values were all small (see Figure 14 on page 61) versus all large density values (see Figure 13 on page 60). However, large variations in the *true* parameter value were found to induce large variations in the density samples, making it impossible to establish a threshold that would consistently distinguish between small and large density values. Therefore, decision logic based on the *magnitude* of the density data was abandoned in lieu of decision logic based on the relative *variations* of the density data. The examples discussed in these last two paragraphs make it clear that measure M_5 by *itself* is not sufficient for making movement decisions, and Section 3.1.4.1 will describe a combined logic based on measures M_2 and M_5 .

The delta density values are normalized via division by M_1 (the maximum density sample) to prevent the large magnitude variations encountered with measure m_2 . Once normalized, the delta

density measure provides consistent indications that the filter-assumed parameter values are either very close to or very far from the true parameter values.

The relative terms of large and small used above require a more precise definition. A simple threshold-based logic is given by

$$\text{If } M_5 > T_1 \Rightarrow \text{variations are sufficiently large} \Rightarrow \text{a move is desired} \quad (87)$$

The threshold, T_1 , is determined empirically in the following way. First, develop the system models in software so that computer simulations can be conducted on a variety of case studies. The case studies should represent the designer's best depiction of real-world variations of the true parameter values. Next, run simulations and collect raw density data and indications of appropriateness of bank motion at each time for these case studies. Post-process the raw data to calculate the measures, and a reasonable choice for T_1 that leads to desired bank movements should be apparent. Validate the decision logic by performing the threshold check in Equation (87).

Counted Filters: This measure provides the number of density samples within a scale factor, γ , of the maximum density sample, indicating if the filter-assumed parameter values are too finely or too coarsely discretized.

$$M_6 = \sum_{j=1}^J \xi(j) \quad \text{where} \quad \xi(j) = \begin{cases} 1, & f_{z|a,z}(j) > \frac{M_1}{\gamma} \\ 0, & \text{otherwise} \end{cases} \quad (88)$$

Specifically, a filter is considered "counted" $\Rightarrow \xi(j) = 1$ if the elemental filter's density sample, $f_{z|a,z}(j)$, is greater than the maximum density sample, M_1 , divided by γ . This is illustrated in Figure 17 where 7 out of 9 filters have density samples greater than M_1/γ . For $\gamma = 10$, the measure is indicating how many of the density samples are within an order of magnitude of the maximum density sample. Notice that M_6 will have an integer value in the range $[1, J]$ and leads to two possible

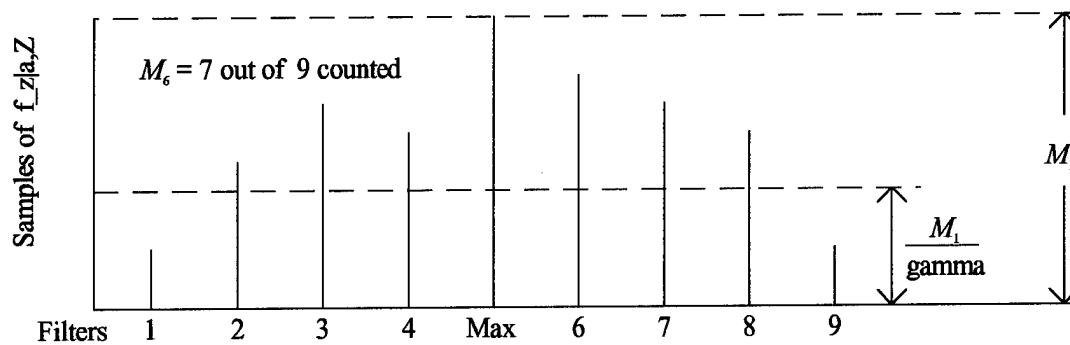


Figure 17. Illustration of Scale Factor in Calculating Measure M_6

decisions. If the filter-assumed parameter values are too finely discretized about the true parameter values, as shown in Figure 18, then the filters will all tend to have density samples of the same magnitude and thus M_6 will be large, i.e., close to the number of filters J . This would warrant an expansion of the bank. The potential flaw in this logic occurs when all the density samples are small but still of the same magnitude, as in Figure 14 on page 61. Clearly a movement would be better than an expansion; so it will be necessary to check first for movement of the bank and give this precedence over an expansion. This is illustrated in the flowchart presented in Section 3.1.4.

The reason for giving bank movements precedence over bank expansions is arbitrary and lies primarily in the development of the decision-making process. It is necessary to establish a structure for the decision-making process that assumes either moves or expansions have precedence. This prevents the logic from using a measure such as M_6 to cause confusion and simultaneously invoke both a move *and* expansion when only one or the other is most desirable. The choice is arbitrary, and the algorithm could have been constructed with precedence given to expansions and contractions over movements, but this development did not.

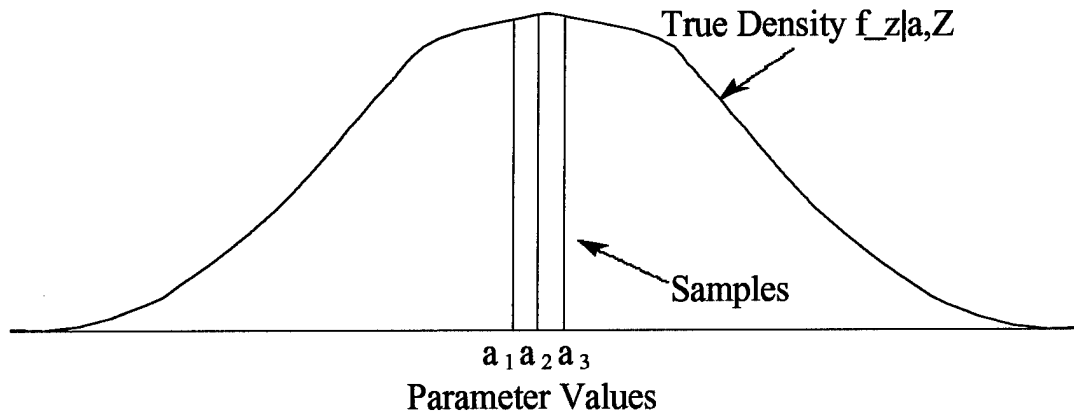


Figure 18. Finely Discretized Filters \Rightarrow Expand the Bank

Alternatively, if the filter-assumed parameter values are too coarsely discretized about the true parameter value, as shown in Figure 19, then all the filter density samples relative to the maximum sample will tend to be small (other than the maximum itself) and thus M_6 will be small, i.e., close to one. This would warrant a contraction of the bank. As with measure M_5 , the relative terms of large and small need further definition. Two threshold based decisions are given by

$$\text{IF } M_6 < T_2 \Rightarrow \text{too coarsely discretized} \Rightarrow \text{contract} \quad (89)$$

$$\text{IF } M_6 > T_3 \Rightarrow \text{too finely discretized} \Rightarrow \text{expand} \quad (90)$$

Additionally, measure M_6 has a second degree of freedom in the selection of the scale factor γ . The same methodology discussed below Equation (87) for the selection of threshold T_1 should be employed here for γ and thresholds T_2 and T_3 . Empirically determining all these values may seem intimidating, if not tedious, at first. However, some simple guidelines should assist considerably in this effort. First, T_3 must be greater than T_2 ($T_3 > T_2$), otherwise the logic could attempt to contract and expand the bank at the same time, which makes no sense. Second, T_2 should be close to one; since a contraction is only needed when all but one or two of the filter-assumed parameter values

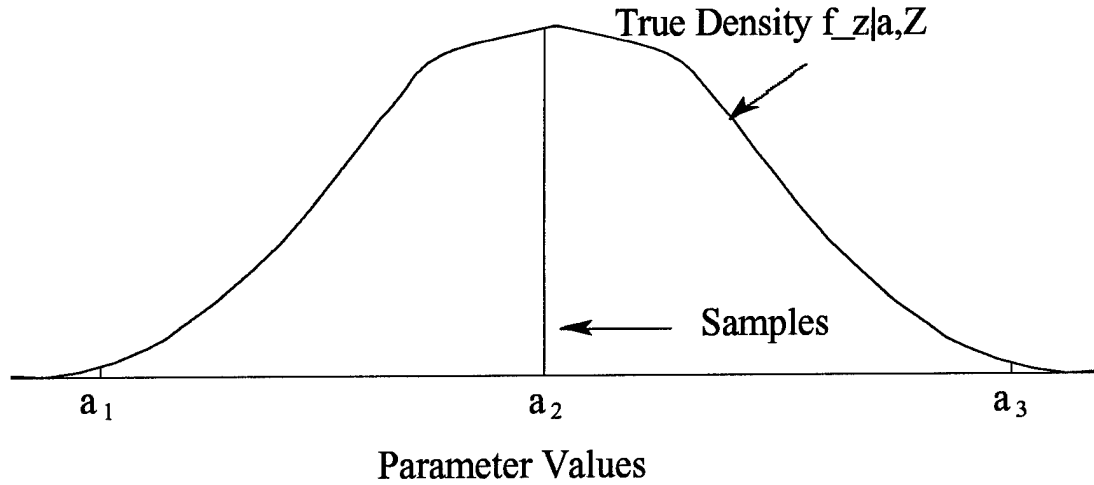


Figure 19. Coarsely Discretized Filters \Rightarrow Contract the Bank

are far from the true parameter values. The exception to this rule is found when a large number of filters (more than 7) are dedicated to a single dimension of the parameter estimates, in which case T_2 may need to be greater than one or two. Third, T_3 should be close to J (the number of filters in the bank) since an expansion is only needed when most, if not all, of the filter-assumed parameter values are clustered around the true parameter values. Finally, it is best to set both T_2 and T_3 prior to selecting the scale factor γ ; since this last degree of freedom is much more problem-dependent than the thresholds, which are mainly affected by the number of filters in the bank. Analysis of the raw measure data will give a reasonable indication of where to set γ such that contractions and expansions occur when desired.

Bad Filters: This measure represents the number of “bad” filters, “bad” in terms of a large likelihood quotient $\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$ for each filter. Recall that a large likelihood quotient implies the filter is a poor match to truth, and by simply counting the number of bad filters, decisions can be made concerning the appropriate harshness of an intended bank movement. Thus let measure M_7 be

defined as:

$$M_7 = \sum_{j=1}^J \xi(j) \quad \text{where} \quad \xi(j) = \begin{cases} 1, & \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j > T_4 \\ 0, & \text{otherwise} \end{cases} \quad (91)$$

A forthcoming discussion on the types of bank movements employed will show the utility of this measure. The basic concept is that, if all the filters are a bad match to truth *and* a decision to move the bank has already been made, then the harshness of this move should be hard versus medium or soft. These relative terms for the harshness of a move indicate how far the bank is moved in the parameter space and will be explained in Section 3.1.4.2.

An alternative use of the information contained in this measure was considered but not thoroughly pursued due to time constraints. Notice that $\xi(j)$ in measure M_7 associates a binary value with each elemental filter, and the concatenation of these binary values could be viewed as a binary word. Consider the three cases [1 0 0 0 1], [1 1 0 0 0], and [0 0 0 1 1] which all result in $M_7 = 2$, but depict significantly different scenarios requiring different moving-bank decisions. In this context, the value of M_7 is less important than the location of the 1's and 0's or the decimal value associated with each binary word. For example, large binary words such as $[1 1 0 0 0]_2 = 24_{10}$ would indicate a need to move right, away from the "bad" filters, versus small binary words such as $[0 0 0 1 1]_2 = 3_{10}$ used to indicate a need to move left. This same type of binary word construct could be used to enhance the information content of any measures expressed as sums in this section, i.e., M_5 through M_9 . A recommendation is made in Chapter 5 to pursue a decision-making process that exploits this type of information.

Determination of the threshold, T_4 , is a key issue that warrants additional discussion. The fact that the density function of the quadratic $\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$ for the case in which the filter is a perfect match

to truth can be fully characterized, lends insight into what threshold value will separate the good filters from the bad ones. Recall from the discussion in Section 2.1.2 that, if the filter model matches the truth model, then its residuals will be a zero-mean white Gaussian process with known residual covariance, \mathbf{A}_j . Furthermore, the quadratic form,

$$\chi_j = (\mathbf{r}_j - \mathbf{0})^T \mathbf{A}_j^{-1} (\mathbf{r}_j - \mathbf{0}) \quad (92)$$

which explicitly involves both the mean and covariance of the Gaussian residuals, is Chi-Squared distributed [65]. The well-known density function for χ_j has a single degree of freedom equal to the dimension of the residuals, m , and is given by:

$$f_\chi(c) = \frac{1}{\Gamma(m/2)2^{m/2}} c^{(m/2)-1} e^{-c/2}, c \geq 0 \quad (93)$$

where c is a dummy variable associated with the random variable χ and Γ is the standard Gamma function. Given this knowledge of the density function associated with “good” filters, a threshold can be chosen based on the probability of correctly classifying “good” filters. Express the probability that the Chi-Squared variable will lie below a threshold as

$$P_\chi(\chi_j \leq T_4) \quad (94)$$

The shaded area in Figure 20 represents this probability for an example with $m = 6$ and $P_\chi = 0.995$. This is the probability of correctly classifying “good” filters, which if available as a design parameter, should be used to set the threshold T_4 . One simple method for finding T_4 given a desired P_χ is to utilize a MATHCAD function “ $T_4 = qchisq(P_\chi, m)$ ” which is a function of both the desired probability and the dimension of the residuals [39].

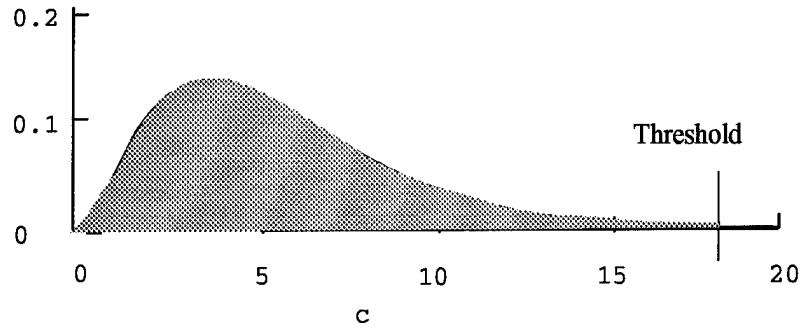


Figure 20. Chi-Squared Density for $m=6$ Degrees of Freedom

A more ideal approach to choosing the threshold would incorporate false and missed alarm rates and a likelihood ratio of the form:

$$l(\chi) = \frac{f_{\chi}(c|H_1)}{f_{\chi}(c|H_0)} \quad (95)$$

where $f_{\chi}(c|H_i)$ represents the density function of the random variable χ given that hypothesis H_i is true. If the hypotheses are established as

$$\begin{aligned} H_0 &\Rightarrow \text{filter perfectly matches truth} \\ H_1 &\Rightarrow \text{filter is mismatched to truth} \end{aligned}$$

then the density function $f_{\chi}(c|H_0)$ is precisely the Chi-Squared density function given by Equation (93) and the *generalized* Chi-Squared density function, $f_{\chi}(c|H_1)$, must be determined. Given that a closed form expression could be found for $f_{\chi}(c|H_1)$, then the common likelihood ratio tests such as the Neyman-Pearson Test or a Sequential Probability Ratio Test could be employed as described in [12, 65] and Section 2.1.3. These tests return threshold values based on false and missed alarm rates specified by the designer. Recall the definition of a false alarm rate, α , as the probability of

incorrectly declaring hypothesis H_1 is valid when H_0 is true. Define a missed alarm rate, β , as the probability of not declaring hypothesis H_1 is valid when H_1 is true. Considerable effort was given to finding a closed form for $f_X(c|H_1)$, resulting in expressions for the first two moments of the random variable. The derivations of these moments are presented in Appendix A. However, a general equation for the density function was not found and is a topic for future research. The research in this area branched off to numerical techniques for calculating the probability, P_X , for the *generalized* Chi-Squared case, leading to the new discretization technique presented in Section 3.2. In summary, the method currently used to set threshold, T_4 , is to designate the probability of correctly classifying “good” filters, $P_X(\chi_j \leq T_4)$, then use MATHCAD [39] to calculate T_4 via $T_4 = qchisq(P_X, m)$.

Similar Filters: The fundamental concept of measure M_7 , which focused on distinguishing between good and bad filters, led to the idea of measure M_8 . “Similar filters” refers to filters having similar $\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$ values. If a filter’s likelihood quotient is close to the likelihood quotient associated with the filter having the maximum probability weight (indicated by measure M_4), then that filter is declared similar to the one with the maximum probability weight. This is determined by taking the ratio of the likelihood quotients $L_j = \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$ and $L_{M_4} = \mathbf{r}_{M_4}^T \mathbf{A}_{M_4}^{-1} \mathbf{r}_{M_4}$, then checking if this ratio falls within the region defined by $1 \pm T_5$. If the ratio falls within this region, then the filters are similar. The measure M_8 is given by:

$$M_8 = \sum_{j=1}^J \xi(j) \quad \text{where} \quad \xi(j) = \begin{cases} 1, & 1 - T_5 < \frac{\mathbf{r}_{M_4}^T \mathbf{A}_{M_4}^{-1} \mathbf{r}_{M_4}}{\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j} < 1 + T_5 \\ 0, & \text{otherwise} \end{cases} \quad (96)$$

Note that, in general, the filter having the maximum probability weight is not guaranteed to have the minimum likelihood quotient, but the ratio $\frac{L_{M_4}}{L_j}$ provides insights into the similarity or in-

distinguishability of the filters. The likelihood quotient, L_{M_4} , would typically be less than any other L_j value since the filter associated with M_4 is currently selected by the MMAE as the best match to truth. However, with the measurement noise covariance, \mathbf{R}_j , as the parameter, $\mathbf{A}_j = \mathbf{H}_j \mathbf{P}_j^- \mathbf{H}_j^T + \mathbf{R}_j$ is heavily influenced by \mathbf{R}_j when the bank is far from truth (as in the case of medium and hard moves being desired). Therefore, $L_j = \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$, which utilizes \mathbf{A}_j^{-1} , will tend to be smaller than L_{M_4} for filters assuming $\mathbf{R}_j > \mathbf{R}_t$ and larger than L_{M_4} for filters assuming $\mathbf{R}_j < \mathbf{R}_t$ (excluding the case when $L_{M_4} = L_j$). This motivates the symmetric threshold tests shown in Equations (96) and (97). The motivation for using L_{M_4} in the ratio versus the *minimum* L_j value is that many of the decisions used in this algorithm (soft moves, contractions and expansions) are centered around the filter associated with M_4 .

The threshold, T_5 , must be determined empirically using the same method discussed below Equation (87) for the selection of threshold T_1 . This measure indicates a lack of variation in the filter models, and by simply counting the number of similar filters, decisions can be made concerning the appropriate harshness of an intended bank movement. A sequential set of decisions have been made (with sequential precedence given to move decisions and *then* expansion/contraction decisions, as discussed previously), indicating a desire to move the bank as illustrated in the flowchart in Section 3.1.4. The basic concept is that, if all the filters are a similar match to truth *and* a decision to move the bank has already been made, then the harshness of this move should be hard versus medium or soft. This need for a hard move is best illustrated in Figure 14 on page 61, in which all the filters will tend to have similar likelihood quotients. This logic mimics that used with measure M_7 and, at first look, it appears that the measures are redundant. However, empirical analysis indicated that both are necessary to account for the various test cases used in this study. In other words, given that all the filter models are far from the truth model, it is sometimes easier to detect that all the filters

are bad and sometimes easier to detect that all the filters are similar to each other. The forthcoming flowchart will indicate that a logical OR allows these two measures to be combined.

Moderately Similar Filters: The last measure is almost identical to measure M_8 , and its use is very problem-dependent. In this case, the number of “moderately similar filters” in terms of moderately similar likelihood quotients for each filter is determined.

$$M_9 = \sum_{j=1}^J \xi(j) \quad \text{where} \quad \xi(j) = \begin{cases} 1, & 1 - T_6 \leq \frac{\mathbf{r}_{M_4}^T \mathbf{A}_{M_4}^{-1} \mathbf{r}_{M_4}}{\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j} \leq 1 + T_6 \\ 0, & \text{otherwise} \end{cases} \quad (97)$$

This measure simply allows the designer to incorporate medium sized moves rather than being limited to hard and soft moves (see discussion in Section 3.1.4). The basic concept parallels that used with measure M_8 in that, if all the filters are a moderately similar match to truth *and* a decision to move the bank has already been made, then the harshness of this move should be medium versus soft. This flexibility was found useful for the test cases used in this research. The threshold, T_6 , must be determined empirically (see methodology on page 70) but will always be greater than T_5 . If T_6 is less than T_5 ($T_6 < T_5$), then a given filter could fall in the counterintuitive category of being deemed “similar” without being deemed “moderately similar”.

3.1.4 Algorithm Description

The previous sections introduced the information exploited by the density algorithm and introduced some of the logic used for decision making. This section will focus on the logic flow of the decisions, beginning with the methods exploiting the information contained in measures $M_1 - M_6$ and explained under the *Main Routine* subsection. These measures are needed to make the first level of the moving-bank decisions such as move left, move right, expand, and contract. This discussion is followed by the *Move Subroutine* subsection, which explains the use of measures $M_7 - M_9$

needed to determine the appropriate harshness of an intended bank movement. Each major event or decision shown in the flowcharts is labeled with a number to help the reader relate the discussion below with the figures. A flowchart is not presented for the subroutine *Redistribute* since the logic flow is quite simple and a written discussion should suffice.

3.1.4.1 Main Routine

The main routine is depicted in Figure 21 and by virtue of the order of the decision blocks (diamond-shaped blocks), movements are given order precedence over contractions and expansions. Recall the discussion on page 71 which explained that the process used to develop the algorithm led to giving bank movements precedence over contractions and expansions. This choice is nothing more than a judgment made by the author and not based on any criteria for optimality, nor is it considered necessarily superior to the method of giving contractions and expansions precedence over movements. Simply stated, a determination was made to proceed with the decision structure illustrated in Figure 21 recognizing that other methodologies were equally viable options.

1. *Form Measures*: Calculate measures $M_1 - M_6$ of Section 3.1.3 used in the main routine.
2. *Check for Move Left*: The logic used here combines two measures with a logical AND, along with a check that the bank is not already at the left end of the parameter space. Consider the situation depicted in Figure 22, where a move to the left is desired. The filter associated with the maximum density value is at the left edge of the bank $\Rightarrow M_2 = 1$. Also, considerable variation of the density samples exists $\Rightarrow M_5 > T_1$. Finally, the parameter value associated with filter 1 is not equal to the minimum allowable value for the parameter ($a_1 \neq a_{\min}$). All three requirements must be met to deem that a movement to the left is needed, as shown by the following IF-THEN statement:

$$\text{IF } (M_2 = 1 \ \& \ M_5 > T_1 \ \& \ a_1 \neq a_{\min}) \ \text{THEN } (\text{Move Left}) \quad (98)$$

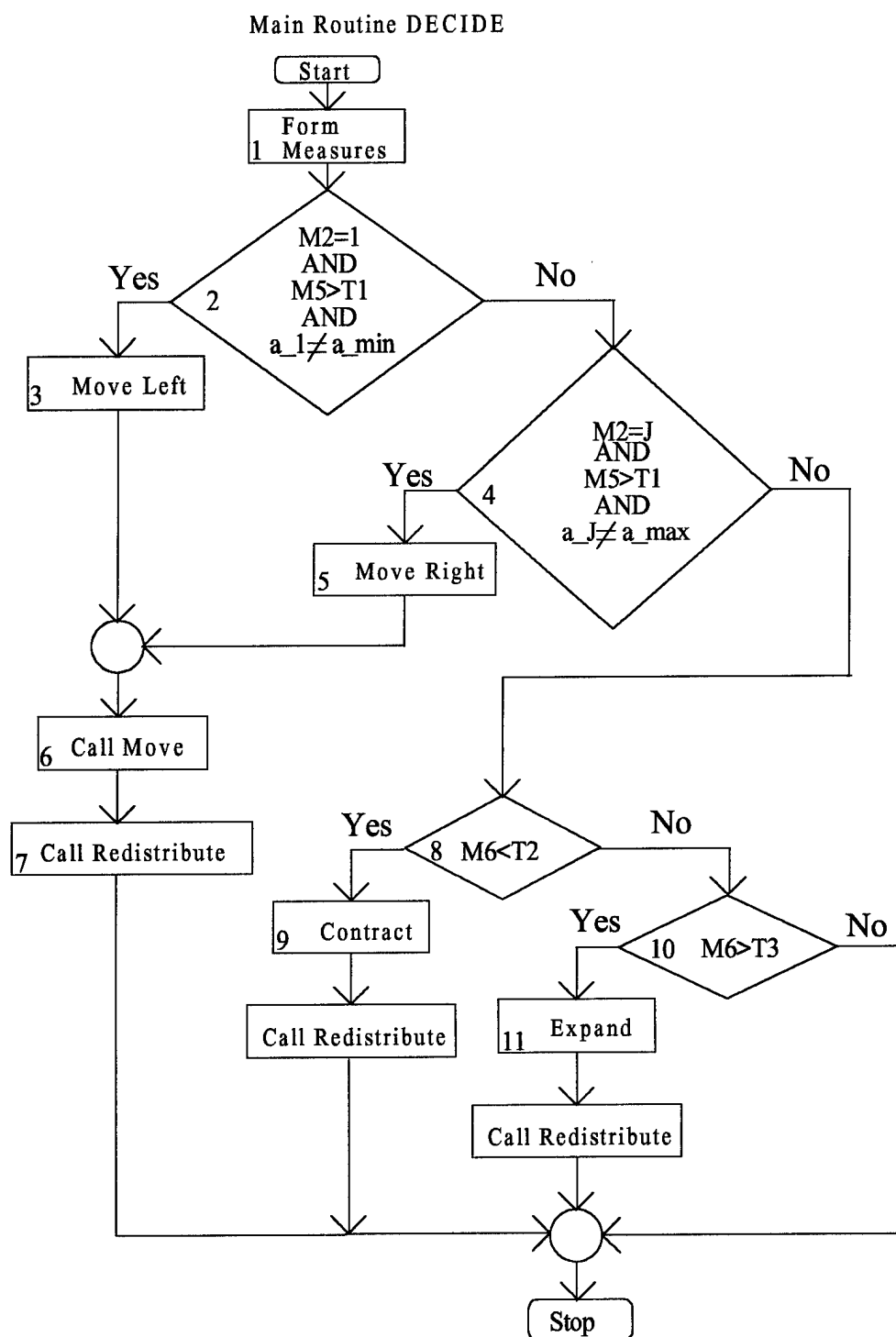


Figure 21. Density Algorithm Main Routine

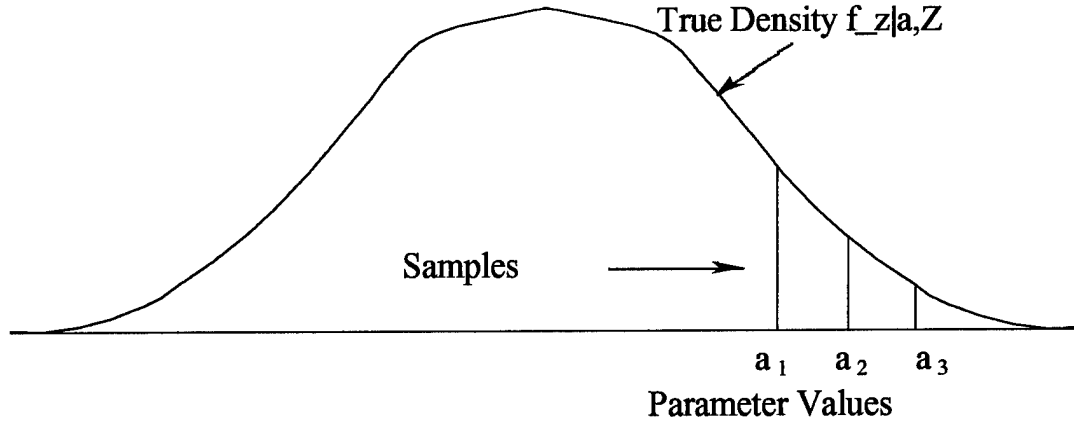


Figure 22. Move MMAE Bank to the Left

A decision to move the bank based solely on measure M_2 or M_5 would be victim to some special cases. First, the true parameter value may be centered in the filter bank (in terms of the filter-assumed parameter values), but a single noise sample could cause one of the filters on the edge of the bank to become the filter with the largest density sample momentarily, i.e., $M_2 = 1$ or J . A movement would not be desired based solely on measure M_2 , and by also recognizing that the variation of the density samples is insufficiently large, an undesirable movement would be prevented under such a circumstance. Second, measure M_5 does not indicate which direction to move the bank, but simply that a move is needed. The direction information is found in measure M_2 , leading to the natural combination of these two measures.

3. *Set Action to Move Left*: A flag is set indicating a move to the left; so the subroutine *Move*, which is about to be called, will know which direction to move the bank.

4. *Check for Move Right*: The concepts and ensuing logic are identical to those for decision block 2, *Check for Move Left* with the following exceptions. First, the filter associated with the maximum density sample must be at the right edge of the bank $\Rightarrow M_2 = J$. Second, the parameter

value associated with filter J must not be equal to the maximum allowable parameter value ($a_J \neq a_{\max}$). The IF-THEN is stated as

$$\text{IF } (M_2 = J \ \& \ M_5 > T_1 \ \& \ a_J \neq a_{\max}) \ \text{THEN } (\text{Move Right}) \quad (99)$$

5. *Set Action to Move Right*: A flag is set indicating a move to the right; so the subroutine *Move*, which is about to be called, will know which direction to move the bank.

6. *Call Move*: Call the subroutine *Move*, which will be discussed in detail presently.

7. *Call Redistribute*: A move, expansion or a contraction will establish a new set of *potential* $[a_1, a_2, \dots, a_J]$ values in parameter space. Given that one of these three operations has been executed, call the subroutine *Redistribute* to ensure that the newly assigned parameter values for the filters lie within the admissible region of the parameter space. It may seem unlikely that a bank contraction would lead to parameter values outside the admissible parameter region. However, the method used to contract the bank may lead to a simultaneous move left or right, since the bank is centered on the parameter value associated with the filter having the maximum probability. This is discussed further under item 9, *Contract the Bank*. Similarly, item 11, *Expand the Bank*, explains that a bank movement may be induced by a bank expansion. One obvious alternative would be to perform *pure* contractions and expansions about the current center of the bank (with movements taking place *only* when dictated by a completely separate move logic), eliminating any bank movement.

The newly assigned parameter values for the left and right ends of the bank (a_1 and a_J) are simply compared to the minimum and maximum allowable parameter values (a_{\min} and a_{\max}) respectively. If either newly assigned endpoint falls outside the admissible parameter space, then it is assigned the appropriate value of a_{\min} or a_{\max} . This relatively simple operation is depicted in Figure 23. Notice that the breadth of the parameters spanned by the bank is maintained in the redistribution process. The parameter breadth is defined as the difference between the maximum and

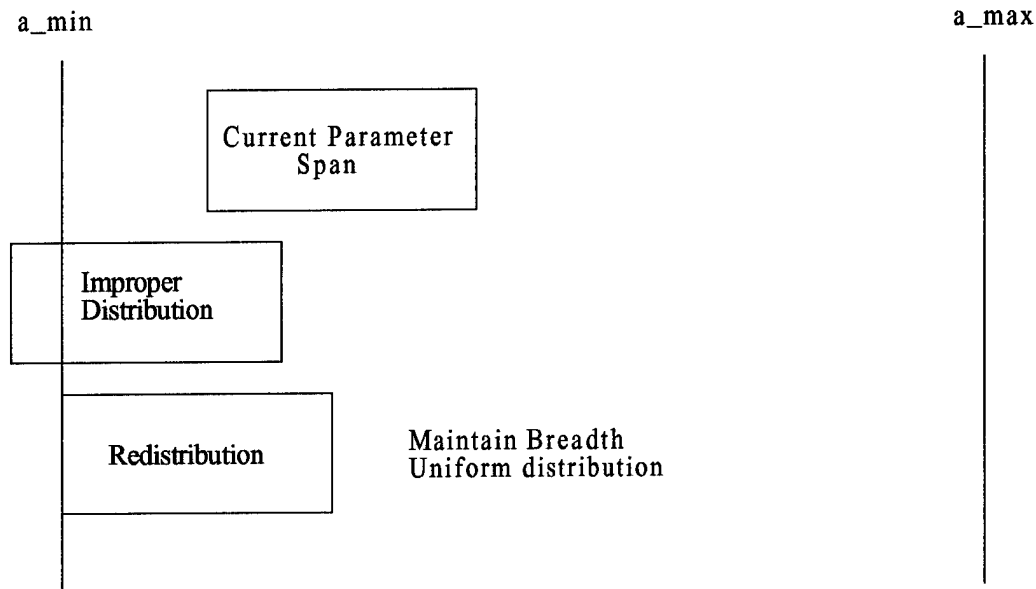


Figure 23. Redistribute Filter Bank

minimum filter-assumed parameter values in each dimension.

$$\text{Breadth} = a_J - a_1 \quad (100)$$

This implies that all the $[a_1, a_2, \dots, a_J]$ values are redistributed (moved left or right) by the same amount rather than adjusting only those parameter values that have fallen outside the admissible parameter space or using some other criteria to rediscritize all the parameter values within the bank. Simply adjusting only those parameter values that have fallen outside the admissible parameter space would result in a contraction of the bank, which is not warranted at this time. Rediscrizing the parameter values based on other criteria is employed by various versions of the density algorithm, including the density algorithm combined with an on-line Sheldon discretization described in Section 3.1.5.4 and the density algorithm combined with the probability based discretization method

described in Section 3.3. The relative merits of the *ad hoc* method presented here versus the discretization methods based on other criteria are discussed in Chapters 4 and 5.

8. *Check for Contraction*: Note in Figure 21 that this check is *only* performed if checks for both leftward and rightward moves have negative results. Recall the discussion in Section 3.1.3 which explained that, if the filter-assumed parameter values are too coarsely discretized about the true parameter values, then all the density samples relative to the maximum sample will tend to be small and thus M_6 will be small, i.e., the number of counted filters will be close to one. This would warrant a contraction of the bank, as shown in Figure 24. Measure M_6 is simply compared to threshold value T_2 , and if M_6 is less than T_2 ($M_6 < T_2$), then a contraction is performed.

9. *Contract the Bank*: Notice in Figure 24 that the bank is contracted about the parameter values associated with the filter having the maximum probability, i.e., centered around M_4 . The idea is that this filter's assumed parameter values provide the best guess, among those in the bank, for the true parameter values. Therefore, it is the logical choice for the center of the bank in parameter space. This is no guarantee that this method of contraction, which is symmetric about the filter associated with M_4 , is optimal. Alternatively, if M_4 is not in the center of the current filter bank (prior to contraction), then the asymmetry that exists in the current bank could be applied to the newly contracted bank. However, applying this asymmetry does not guarantee optimality either. Therefore, the symmetric method of centering around M_4 was chosen for simplicity. The next step is to determine the remaining parameter values for the filters within the bank by first determining the amount of contraction desired.

The underlying concept for choosing the new filter-assumed parameter values in the bank is based on human eyesight. Specifically, it is desirable to give the MMAE both a foveal and peripheral view of the parameter space. The foveal view would consist of several filters (say 3 out of 5 filters) focussing on the current best estimate of the parameter values, as shown by example in Figure 25.

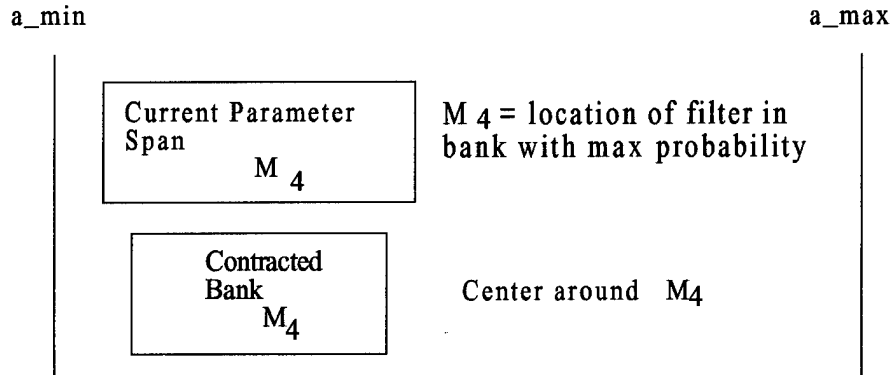


Figure 24. Contraction of Filter Bank

This is accomplished by centering around the maximum probability filter as mentioned above. Since the foveal view filters would be relatively close to each other with respect to their assumed parameter values, these filters would tend to have density samples that are close in magnitude and contribute to the number of “counted” filters via measure M_6 (recall Equation (88) and Figure 17). Furthermore, the maximum density sample is most likely to be associated with one of these foveal filters. This motivates contracting the bank such that the number of counted filters consistently matches the number of desired foveal filters. The peripheral view of the MMAE bank would consist of the remaining filters (say 2 out of 5 filters) which would assume parameter values far from the current best parameter estimate. These filters will not be counted via M_6 since their associated density samples will be small compared to the maximum density sample. These peripheral filters are lying in wait for a dramatic change of the true parameter values and give the bank the flexibility to react to such a change. Apply these thoughts to the following example. Let $J = 5$ and assume Figure 26 represents the density samples associated with the five filters in the bank prior to a desired contraction. Notice that the number of significant or “counted” densities is one, $M_6 = 1$. Now let the bank undergo a

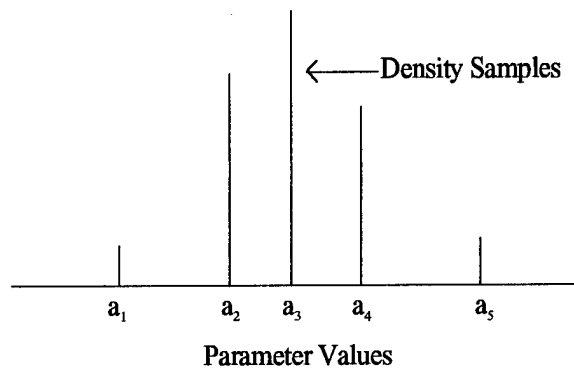


Figure 25. Filter Bank After Contraction

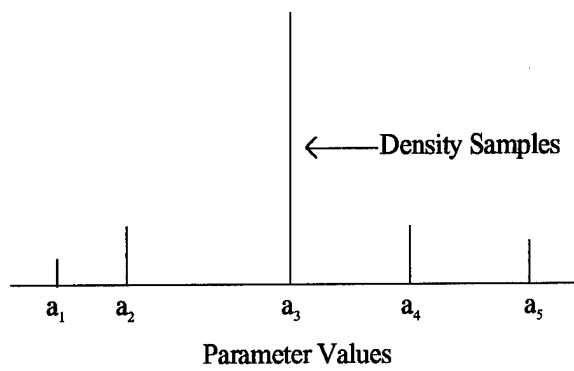


Figure 26. Filter Bank Prior to Contraction

contraction followed by the propagate and update cycles of the Kalman filters. The contrived density samples for this example might appear as in Figure 25 where the number of “counted” filters has increased from $M_6 = 1$ to $M_6 = 3$. Based on these observations, the following method was developed which allows the designer to choose the number of desired foveal or counted filters.

Contraction will be applied to the filters in a uniformly spaced parameter range, where the uniform spaced parameter range refers to the integer numbering assigned to the filters (1, 2, 3,...J). The spacing here is clearly uniform despite the highly nonuniform spacing in the actual parameter space that may appear between the parameter values assumed by the filters. This is easier to understand by looking at the contrived example shown in Table 1, in which the parameter values are not uniformly spaced but the numbers assigned to the filters obviously are uniformly spaced. Recall that a contraction decision is based on measure M_6 which takes on integer values in the range [1,J]. Therefore, it makes sense to work in the uniform parameter space when determining if a contraction is needed *and* to determine the amount of contraction. Additionally, from a design standpoint, it makes sense to designate the number of desired filters (an integer value) in the foveal and peripheral views of the MMAE.

Table 1. Uniform and Nonuniform Parameter Values

Filter #	1	2	3	4	5
Parameter Value	5	10	50	120	150

Define a contraction factor, κ , equal to the desired number of counted (foveal) filters and applied in the uniform parameter space to determine new uniform parameter values for the filters, which are then converted to actual filter-assumed parameter values via interpolation or extrapolation described presently. As previously stated, the new contracted uniform values, a_u , are found by centering around the filter-assumed parameter value having the maximum probability using the

following equation:

$$\begin{aligned} \mathbf{a}_u^T &= [a_{u1}, a_{u2}, \dots, a_{uJ}] = [M_4, M_4, \dots, M_4] + \\ &\quad \kappa \cdot \left[-\frac{J-1}{2(J-1)}, -\frac{J-3}{2(J-1)}, -\frac{J-5}{2(J-1)}, \dots, \frac{J-J}{2(J-1)}, \dots \right. \\ &\quad \left. \dots, \frac{J-5}{2(J-1)}, \frac{J-3}{2(J-1)}, \frac{J-1}{2(J-1)} \right] \end{aligned} \quad (101)$$

The first bracketed term in Equation (101) is just a J-dimensional vector with each element equal to M_4 . The second bracketed term in Equation (101) will always have the form $[-\frac{1}{2}, \dots, 0, \dots, \frac{1}{2}]$, resulting in minimum and maximum uniform parameter values of $a_{u1} = M_4 - \frac{\kappa}{2}$ and $a_{uJ} = M_4 + \frac{\kappa}{2}$ respectively. Therefore, the new uniform parameter breadth is given by $a_{uJ} - a_{u1} = (M_4 + \frac{\kappa}{2}) - (M_4 - \frac{\kappa}{2}) = \kappa$ and supports the design goal of maintaining a set number of counted (foveal) filters. For example, the case of $J = 5$, $M_4 = 4$ and a desired foveal view of 3 filters,

$$\mathbf{a}_u^T = [4, 4, 4, 4, 4] + 3 \cdot \left[-\frac{1}{2}, -\frac{1}{4}, 0, \frac{1}{4}, \frac{1}{2} \right] = [2.5, 3.25, 4, 4.75, 5.5]$$

Notice that for this example, the newly contracted parameters only span a uniform range of $5.5 - 2.5 = 3$ (equal to κ) versus the original span of $5 - 1 = 4$ (equal to $\kappa + 1$). Likewise, the actual filter-assumed parameter values found via interpolation and extrapolation will span a contracted range. Notice that the method used here will always result in a contraction of the bank to $\frac{\kappa}{\kappa+1}$ of the original bank span, regardless of the measure M_6 . Initially, consideration was given to incorporating M_6 into the contraction process such that smaller values of M_6 (indicating a *very* coarse discretization about the true parameter values) would induce larger amounts of contraction. However, performance

analysis found that a single level of contraction was sufficient, since the bank rarely required a series of contractions, and this single level of contraction was used for simplicity.

The contraction method just described is used in the final implementation; however, an alternative method is recommended for consideration. Rather than uniformly spacing all of the filters over the newly contracted bank span, one might choose to group the foveal filters tightly in the center of the new span and place the peripheral filters farther to the outside. One such approach is illustrated in Figure 27 for a five-filter example. Notice that the parameter values for the three foveal filters

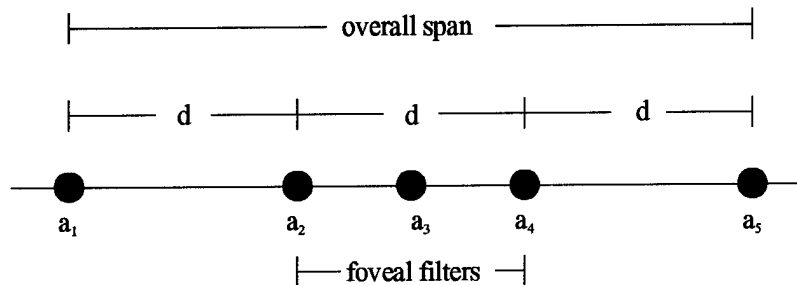


Figure 27. Alternative Spacing of New Parameter Values for Contractions

are evenly spaced over the distance "d" in the *uniform* parameter space and the peripheral filters are spaced a distance "d" from the outer foveal filters. In general, the number of desired foveal filters is simply $J - 2$. Recall that the contraction is performed in the *uniform* parameter space prior to conversion to the actual parameter space, which is discussed in the next paragraph. Therefore, the distance "d" is an integer value equal to the number of desired foveal filters ($J - 2$) and the *overall* span is equal to $3d$. This approach is recommended for its simplicity, and many alternatives are possible.

The next issue is, how to convert from the uniform filter parameter spacing to the actual filter-assumed parameter values. The solution is to plot the uniform filter parameter values versus the actual filter-assumed parameter values, as shown in Figure 28. The contraction will actually take place in the uniform parameter space as described above, resulting in non-integer values within this space. Next employ linear interpolation and extrapolation to determine values in the actual filter-assumed parameter space corresponding to the non-integer values. Consider the following example contrived to illustrate the interpolation process. Specifically, use the filter parameter values shown in Table 1 and assume the contraction process requires a new uniform parameter value of $a_{u1} = 2.5$. The mapping of the uniform parameter values to the actual filter parameter values is shown in Figure 28 including the new filter-assumed parameter value of $a_1 = 30$ (shown as an "0") which corresponds to the new uniform parameter value of $a_{u1} = 2.5$. Similarly, notice for the example presented in the previous paragraph that $a_{u5} = 5.5$, which is outside the current parameter span and requires extrapolation in the uniform parameter space. Linear extrapolation is applied for simplicity and provides adequate results for the relatively short extrapolation distances required here. For instance, the maximum extrapolation distance required in this example is $a_{u5} - J = 5.5 - 5 = 0.5$. Note that this method of contracting the bank may additionally result in a small move to the left or the right. This is caused by centering on the filter-assumed parameter value having the maximum probability unless the associated filter is currently in the center of the bank. The example above illustrates this attribute since centering around $M_4 = 4$ results in a uniform parameter value of $a_{u5} = 5.5$ and thus actual parameter values which are to the right of the current bank.

Finally, a call to the subroutine *Redistribute* is made to ensure the new filter-assumed parameter values lie within the admissible parameter range. The vector \mathbf{a}_u is independently obtained for each dimension of the parameter vector.

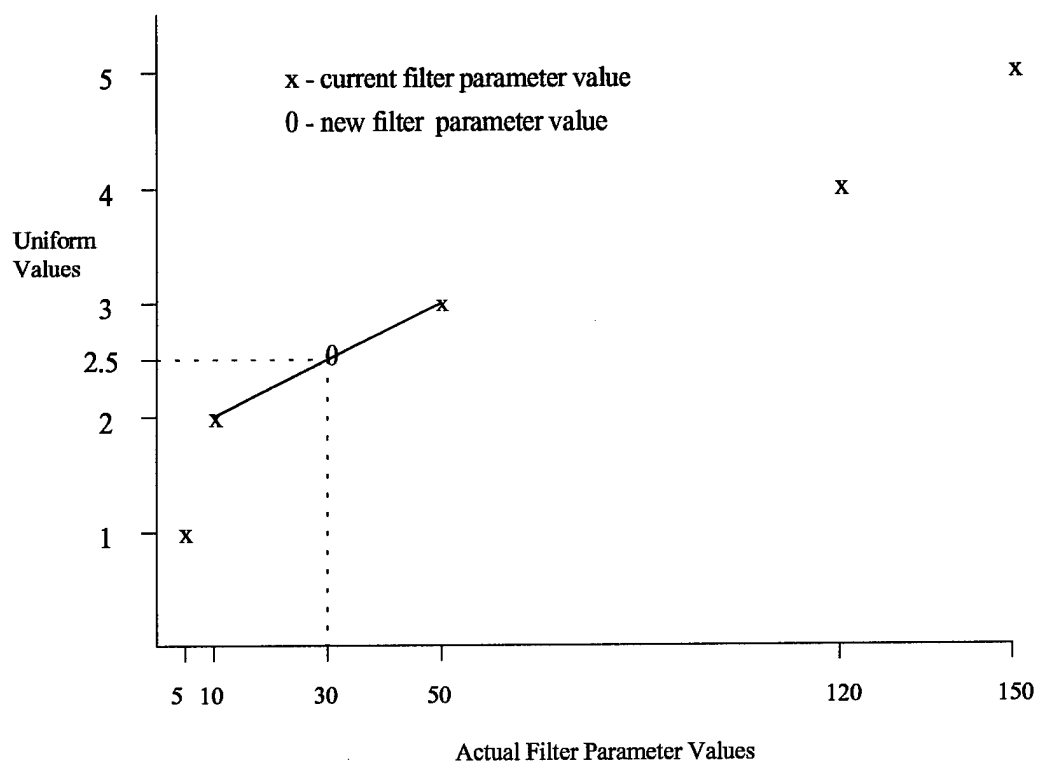


Figure 28. Uniform Versus Actual Filter-Assumed Parameter Values with Interpolation

10. *Check for Expansion*: Recall the discussion in Section 3.1.3, which explained that if the filter-assumed parameter values are too finely discretized about the true parameter value, then all the filters will tend to have density samples of the same magnitude and thus M_6 will be large, i.e., close to the number of filters, J . This would warrant an expansion of the bank as shown in Figure 29. Measure M_6 is simply compared to threshold value T_3 and if $M_6 > T_3$, then an expansion is performed. Also, recall the need to check for a move prior to checking for an expansion (or contraction), as discussed in Section 3.1.3. The sequence of decision blocks shown in the flowchart (Figure 21) ensures the move decisions have precedence over an expansion.

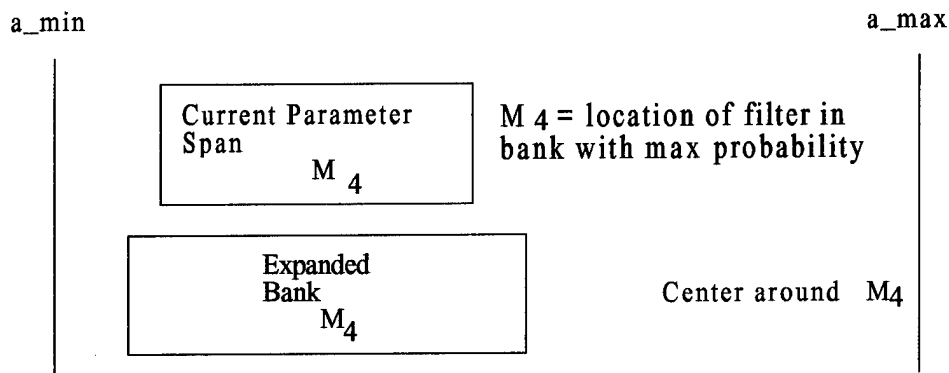


Figure 29. Expansion of Filter Bank

11. *Expand the Bank*: Much of the same rational used to contract the filter bank applies to expanding the bank. The method for expansion is simpler in that conversion from the uniform parameter space to the actual parameter space is not performed. Consideration was given to performing expansions in the uniform space and then converting to the actual parameter space, as done with

contractions. However, this would require extrapolations over potentially large distances in the uniform space, leading to poor results that do not adequately represent the desired expansion.

An expansion factor is simply calculated as

$$\text{Expansion Factor} : \varepsilon = \frac{M_6}{\# \text{ of desired counted (foveal) filters}} \quad (102)$$

where the denominator is now motivated by the designer's will to keep a set number of filters in the foveal view of the moving-bank MMAE, and the numerator is simply the number of filters currently in the foveal view. This ratio shows that if too many (more than desired by the designer) of the filter-assumed parameter values are tightly packed around the true parameter values, then the bank will expand until the desired number of filters reside in the neighborhood of truth. The new filter-assumed parameter values are calculated using

$$\begin{aligned} [a_1^{new}, a_2^{new}, \dots, a_J^{new}] &= [a_{M_4}, a_{M_4}, \dots, a_{M_4}] + \\ &\quad (a_J - a_1) \cdot \varepsilon \cdot \frac{1}{2} \left[-\frac{J-1}{J-1}, -\frac{J-3}{J-1}, -\frac{J-5}{J-1}, \dots, \frac{J-J}{J-1}, \dots \right. \\ &\quad \left. \dots, \frac{J-5}{J-1}, \frac{J-3}{J-1}, \frac{J-1}{J-1} \right] \end{aligned} \quad (103)$$

which is easily understood from the following description. Multiply the current breadth of the bank, $[a_J - a_1]$, times the expansion factor, ε , to obtain an expanded breadth of the bank. Next, multiply by the uniform spacing based on the number of filters and given by

$$\frac{1}{2} \left[-\frac{J-1}{J-1}, -\frac{J-3}{J-1}, -\frac{J-5}{J-1}, \dots, \frac{J-J}{J-1}, \dots, \frac{J-5}{J-1}, \frac{J-3}{J-1}, \frac{J-1}{J-1} \right]$$

to obtain an expanded spacing. Finally, add these products to a vector of the parameter value associated with the filter having the maximum probability, a_{M_4} , to obtain a new set of parameter values for the filters, $[a_1^{new}, a_2^{new}, \dots, a_j^{new}]$, centered around a_{M_4} . As with contractions, this method of expanding the bank may additionally result in a small move to the left or the right.

A call to the subroutine *Redistribute* is made to ensure the new filter-assumed parameter values lie within the admissible parameter range. The J-dimensional vector, $[a_1^{new}, a_2^{new}, \dots, a_j^{new}]$, is independently obtained for each dimension of the parameter vector. If only one dimension of the parameter space (or at least not all of the dimensions) requires an expansion, then it seems unreasonable to expand every dimension of the parameter space. Therefore, systematically checking and implementing expansions in each dimension independently provides the flexibility needed to expand in the appropriate dimensions. Additionally, this method is straightforward and a reasonable first approach. Alternatives to componentwise expansion decisions in the *original* coordinate system of the parameter space might also be pursued (i.e., rotate into principal axes and *then* apply componentwise expansion decisions), but this was not done in this research.

3.1.4.2 Move Subroutine

The subroutine *Move* is called by the main routine when a bank movement is warranted. Figure 30 shows the decision making process used in this subroutine. Each major event or decision shown in the flowchart is labeled with a number to help the reader relate the discussion below with the figure.

1. *Form Measures*: Calculate measures $M_7 - M_9$ of Section 3.1.3 used in the subroutine.
2. *Check for Hard Move*: Recall from the discussion of measure M_7 , that if all the filters are a bad match to truth *and* a decision to move the bank has already been made, then the harshness of this move should be hard versus medium or soft. This scenario is handled by checking if $M_7 = J$.

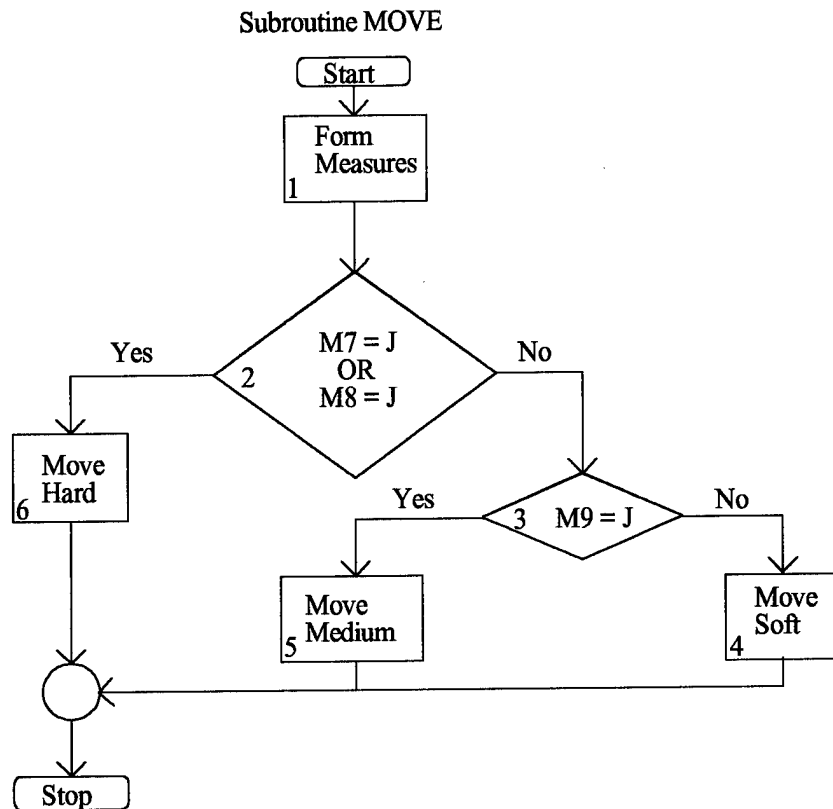


Figure 30. Density Algorithm Move Subroutine

Similarly, recall from the discussion of measure M_8 , that if all the filters are a similar match to truth *and* a decision to move the bank has already been made, then the harshness of this move should be hard versus medium or soft. This scenario is handled by checking if $M_8 = J$. The resulting logic is expressed as

$$\text{IF } (M_7 = J \text{ OR } M_8 = J) \text{ THEN (Move Hard)} \quad (104)$$

3. *Check for Medium/Soft Move*: Recall that measure M_9 allows the designer to incorporate medium sized moves rather than being limited to hard and soft moves. The logic is conditioned on $M_7 \neq J$ AND $M_8 \neq J$ and given by

$$\text{IF } (M_9 = J) \text{ THEN (Move Medium) ELSE (Move Soft)} \quad (105)$$

indicating that, if all the filters have moderately similar likelihood quotients then a medium move would be better than a soft move.

The three types of bank movements are illustrated in Figure 31 as soft, medium and hard right. Moves to the left are performed in the same way. The methods used to determine the new parameter values for the filters are *ad hoc* and based on the concepts described below. In general, these three levels of movement were motivated by analysis of various test cases in preliminary studies/simulations. Initially, only soft moves were invoked, but in cases in which large parameter variations were induced, several soft moves were required and convergence of the filter-assumed parameter values to the neighborhood of the true parameter values was too slow. This motivated the use of hard moves which overcame the slow convergence times. However, some of the case studies induced moderate parameter changes, and a hard move would cause the filter-assumed parameter values to overshoot the true parameter values significantly. Therefore, an intermediate move (medium move) was pursued. Observation of Equations (91), (96) and (97) for measures $M_7 - M_9$ identifies the high level of correlation between these measures (particularly measures M_8 and M_9). Therefore, employing all three levels of moves (or adding more levels) is highly problem-dependent. Empirical

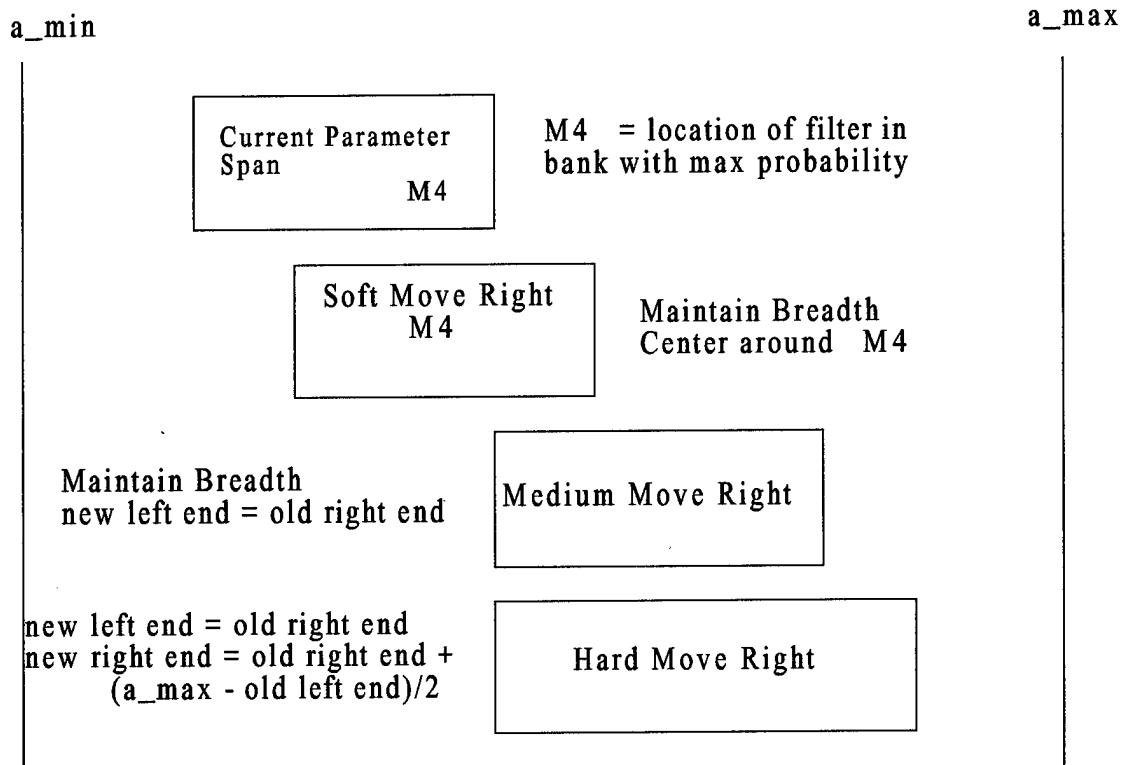


Figure 31. Types of Bank Movements (Soft, Medium, and Hard)

analysis through preliminary studies should indicate if more than one level of movement is needed. If so, employ measure M_7 to discern between two movement levels. If more levels are required, continue adding measures M_8, M_9, M_{10} , (just like M_9 , but with a larger threshold, T_7), and so on to discern between each intermediate level of movement. The only difference among measures M_8, M_9, M_{10}, \dots is the value of the thresholds T_5, T_6, T_7, \dots , which are empirically determined.

Note the use of scalar notation for the parameter values indicating that each dimension of the parameter vector is processed independently. First, it may be necessary to only move in one dimension of the parameter space (or at least some but not all of the dimensions), and working in one dimension at a time permits moves in any or all of the p dimensions. Second, the methods used to determine the new parameter values are easily implemented in a single dimension.

4. *Soft Move*: A soft move is intended to allow the bank to react to small changes in the true parameter value or to hone in on the true parameter value given that it is already within the bank's parameter span. The bank is centered around the filter-assumed parameter value currently having the maximum probability, as this is the best guess, among those in the bank, for the true parameter value at this time:

$$a_{center}^{new} = a_{M_4} \quad (106)$$

where a^{new} represents the new parameter value. The parameter breadth $= [a_J - a_1]$ of the bank is maintained since there is no indication either to expand or to contract the bank. The new parameter endpoints of the bank are determined by placing a filter either side of the new parameter center at a distance of one-half the breadth.

$$a_{1,J}^{new} = a_{M_4} \mp (a_J - a_1) / 2 \quad (107)$$

The remaining filters take on parameter values uniformly distributed between the new center value and the new endpoints. Notice that, under this methodology, there will exist some overlap between the newly placed parameter values and the old parameter values since the center of the new bank

came from the old bank. One alternative to uniformly distributing all the remaining parameter values is to preserve the parameter values in the overlap region and uniformly distributing all the others. This methodology was considered late in the research and time did not permit complete analysis of its performance. It is anticipated that preserving the parameter values in the overlap region would be superior to uniformly distributing all the parameter values, since the preserved filters will not require the initialization needed to “start up” those filters taking on new parameter values (see Section 3.1.5.2) and can avoid transient behavior in their state estimates often encountered by newly initialized filters.

5. *Medium Move*: A medium move allows the bank to make a slightly larger step left or right as the overlap from the old bank to the new bank is reduced to a single filter-assumed parameter value. Specifically, for the move to the right shown in Figure 31, the new left end of the bank takes on the parameter value from the old right end of the bank and visa versa for a move to the left.

$$\begin{aligned} a_1^{new} &= a_J \quad (\text{move right}) \\ a_J^{new} &= a_1 \quad (\text{move left}) \end{aligned} \tag{108}$$

The breadth of the bank is maintained allowing the new right (or left) end of the bank to be calculated as

$$\begin{aligned} a_J^{new} &= a_J + (a_J - a_1) = 2a_J - a_1 \quad (\text{move right}) \\ a_1^{new} &= a_1 - (a_J - a_1) = 2a_1 - a_J \quad (\text{move left}) \end{aligned} \tag{109}$$

The remaining filters take on parameter values uniformly distributed between the new endpoints. Unlike the soft move, there is no overlap of the old and new parameter values except for one endpoint. Therefore, some method such as uniform distribution is required. Other *ad hoc* distribution methods include logarithmic spacing and monitoring estimation accuracy, but only one method is implemented here [23, 49, 68, 69]. Of course, Sheldon’s cost minimization method [68, 69] is not

ad hoc and is utilized through the combination of the density algorithm with Sheldon discretization presented in Section 3.1.5.4.

6. *Hard Move*: A hard move is actually a combination of a move and expansion, as shown in Figure 31. The rationale for this combination is the desire to extend the bank's parameter span over a larger range than that obtained by a medium or soft move, while simultaneously maintaining the single parameter value overlap described for a medium move. This single parameter value overlap is motivated by the conservative nature of the algorithm designer who would rather systematically move right or left toward the true parameter value rather than making huge leaps, resulting in gaps between the old and new filter bank parameter values. In order to satisfy both desires, the bank has to move and expand. The *ad hoc* expansion process described in Section 3.1.4.1 could be used to determine the amount of the expansion associated with the hard move, but an alternative *ad hoc* method discussed here is used and based on the following concepts. Neither method is guaranteed to be superior to the other, and both deserve equal consideration in lieu of some other optimal method.

One endpoint is again governed by Equation (108) to ensure a single filter overlap

$$\begin{aligned} a_1^{new} &= a_J \quad (\text{move right}) \\ a_J^{new} &= a_1 \quad (\text{move left}) \end{aligned} \tag{110}$$

while the other endpoint is a function of the maximum or minimum allowable parameter value (a_{\max} or a_{\min}). Consider the case of a move right, with the understanding that moves to the left are accomplished in an analogous manner (i.e., use a_{\max} versus a_{\min} and swap the a_J 's and a_1 's). One method is simply to set the new right endpoint, a_J^{new} , equal to a_{\max} , resulting in the maximum amount of expansion possible. Preliminary studies indicated that this level of expansion was too severe *for this problem* and resulted in too coarse of a discretization. Another idea is to choose the new right endpoint, a_J^{new} , somewhere between the maximum allowable parameter value, a_{\max} , and the old right endpoint, a_J . This gives the designer control over the amount of expansion by selecting

how close the new right endpoint lies to a_{\max} versus the old a_J . The method used here to select the new right endpoint, a_J^{new} , is to add some fraction of the distance between the old left endpoint, a_1 , and the maximum allowable parameter value, a_{\max} , to the old left endpoint, a_J , as given by

$$\begin{aligned} a_J^{new} &= a_J + (a_{\max} - a_1) / 2 && \text{(move right)} \\ a_1^{new} &= a_1 - (a_J - a_{\min}) / 2 && \text{(move left)} \end{aligned} \quad (111)$$

Notice that, as the bank moves closer to a_{\max} or a_{\min} , the amount of expansion decreases. The fraction of $1/2$ chosen here was arbitrarily determined from preliminary studies. The designer can decrease this fraction ($< 1/2$) to decrease the amount of expansion for the problem at hand or increase this fraction ($> 1/2$) to increase the amount of expansion.

Direct comparison of this expansion method versus the one in Section 3.1.4.1 was not conducted due to time constraints. Additionally, a large expansion could be accomplished in place of the combined hard move / small expansion presented here. However, given an indication that the true parameter value is to the left or right of the current filter-assumed parameter values, a move in the appropriate direction combined with a small expansion makes more sense than simply expanding the bank to a coarse level of discretization. In contrast, if there is no indication of which direction is best to move the bank, then a simple expansion would be better. Since the decision-making logic first checks for a clear indication of an appropriate move direction before resorting to simple large expansions, it accounts for all of these cases.

3.1.5 Performance Enhancements

A short study was conducted to evaluate the performance of the density algorithm. The results of this study motivated the pursuit of several performance enhancements which are detailed in the following paragraphs. It should be noted that none of the enhancements are necessary for the algorithm to function, but some of them were critical for good performance on the example problem used

in this study. These critical enhancements include *Dead Zone for Soft Moves* and a specific form of *Initialization of Newly Declared Filters*, which are anticipated to provide significant improvements in performance for any application of the algorithm. The two other performance enhancements presented in this section include an *ad hoc* method identified as *Decision Delays* and the on-line version of Sheldon's cost minimization algorithm referred to as *On-Line Sheldon Discretization*.

3.1.5.1 *Dead Zone for Soft Moves*

The density algorithm as described to this point made unnecessary soft moves in the presence of true parameter variations. Given that a true parameter changed, the bank would move toward and eventually encompass the true parameter within the bank's parameter span. Once the true parameter was encompassed, an additional soft move might be needed to center the bank on the true value. However, in many cases the algorithm induced multiple soft moves that would overshoot the true parameter and even oscillate about this true value with multiple soft moves to the left and right. The result was erratic parameter estimates that degraded performance. One solution was to implement a dead zone near the center of the bank which precluded a *soft* move if the filter having the maximum probability was within this dead zone. The idea is that, if the bank had converged on the true parameter and was, in fact, nearly centered on this true value, then a filter within the dead zone would have the greatest probability. If this was not the case, then a filter outside the dead zone would have the greatest probability and an additional soft move would be warranted.

The ensuing logic is given by

IF $(\text{Dead1} \leq M_4 \leq \text{Dead2})$ THEN (Don't Make Soft Move)

where Dead1 and Dead2 are integers defining the dead zone and must be determined empirically. If the dead zone is made too wide, such as by placing more than half the filters in this zone, then the algorithm will become sluggish in its convergence to the true parameter. Empirical analysis

showed that the dead zone is more effective if it is biased towards the right side of the filter bank (i.e., larger parameter values) when the parameter of interest is the measurement noise covariance, \mathbf{R} . For instance, with $J = 5$ a dead zone of $\text{Dead1} = 3$ and $\text{Dead2} = 4$ was most effective. This biasing towards larger \mathbf{R} values is justified in the following discussion. Assume the true scalar noise covariance is $R_t = 300$ and the two elemental filters closest to R_t assume parameter values of $R = 100$ and $R = 500$. Both elemental filters are equally spaced above and below truth in terms of the parameter values, but the filter based on the larger assumed value of R would tend to receive a larger probability weight than the filter based on the smaller R value due to the MMAE's tendency to favor the more conservative filter. This phenomenon is illustrated with a scalar measurement example. Recall that the scalar likelihood quotient

$$L_j(t_i) = \frac{r_j^2(t_i)}{A_j} \quad (112)$$

where

$$A_j = \mathbf{H}_j \mathbf{P}_j^- \mathbf{H}_j^T + R_j$$

indicates the “correctness” of the filter with respect to truth, and a large L_j implies that the filter is a bad match to truth. Consider the following examples in which small and large R_j values are referenced with respect to truth:

Example A: R_j small $\Rightarrow A_j$ small $\Rightarrow L_j(t_i)$ large if $r_j^2(t_i)$ is “medium” or “large”

versus

Example B: R_j large $\Rightarrow A_j$ large $\Rightarrow L_j(t_i)$ large only if $r_j^2(t_i)$ is “large”

These examples imply that a much larger residual squared, $r_j^2(t_i)$, is needed to make the filter in example B look bad and receive a smaller probability weight to the same degree as the filter in example A: thus the MMAE's tendency to favor the more conservative filter. The ensuing blending of the parameter estimates given by Equation (51), which is repeated here, will result in a biasing

high of the parameter estimates:

$$\hat{\mathbf{a}}_{\text{MMAE}}(t_i) = \sum_{j=1}^J \mathbf{a}_j p_j(t_i)$$

This concept requires additional research both for validation and for determination of an adequate method to counter its effects, if that is appropriate.

3.1.5.2 Initialization of Newly Declared Filters

The idea of initializing the bank was presented on page 42, which proposes using the blended state estimates as initial conditions for the filters that have newly declared parameter values. Specifically, only the elemental filters not already operational in the bank prior to a move, expansion, or contraction are initialized with $\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+)$. This method is highly recommended to remove unwanted transients that may result if the filters are not initialized with $\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+)$. In particular, consider the alternative in which each elemental filter that has newly declared parameter values chooses to propagate its appropriate state estimate, $\hat{\mathbf{x}}_j(t_i^+)$. This is effectively ignoring the information contained in the blended state estimate, $\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+)$, which is the current best guess for the true state vector. In fact, $\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+)$ is more likely to be in the neighborhood of the true state vector than $\hat{\mathbf{x}}_j(t_i^+)$, particularly since $\hat{\mathbf{x}}_j(t_i^+)$ is based on a parameter value that is currently deemed to be unworthy of being included in the bank. An obvious exception is when a filter assumes parameter values that match truth very well, in which case, this filter is unlikely to have just been assigned a new parameter vector. Clearly, when initializing a Kalman filter, it is best to use an initial condition that is closer to truth than one that is far away, since it will take more sample periods for the filter initialized with a poor initial condition to converge to the true state values.

Recall the IMM approach discussed in Section 2.2.2 which proposed restarting all the filters in the bank after every sample period, given that the probability state transition matrix in the standard

MMAE structure equals identity [5,32]. At first, it may appear that *Initialization of Newly Declared Filters* is nothing more than applying the IMM approach given that the probability state transition matrix equals identity. However, this version of the IMM approach restarts *all* the filters in the bank after *every* sample period, versus restarting only those that have newly declared parameter values. The difference between these two approaches may seem subtle, but given a coarse discretization of the filters in the MMAE, the blended state estimate, $\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+)$, could vary significantly from the elemental state estimates, $\hat{\mathbf{x}}_j(t_i^+)$. Therefore, *re*-initializing an operational elemental filter with $\hat{\mathbf{x}}_{\text{MMAE}}(t_i^+)$ via IMM versus propagating the appropriate value of $\hat{\mathbf{x}}_j(t_i^+)$ could be significant. This version of IMM was not implemented in this research because of its tendency to degrade parameter estimation in lieu of enhanced state estimation.

3.1.5.3 Decision Delays

Decision delays were implemented to reduce erratic bank movement. A simple yet very *ad hoc* method allows the designer to choose the number of sample times that must pass before a new decision (move, expand or contract) can be made. These delays allow the filters to settle out any transients that result from having just taken on new parameter values. A negative aspect of this method is the sluggishness induced into the decision making process when attempting to react to a change in the true parameters. A reasonable process for selecting the amount of decision delay is to simulate test cases with decision delay values of 0, 1, 5 and 10 sample periods. Larger delays may be considered but will likely be too detrimental. Evaluate the performance in each case and select the delay time that provides the best trade-off of parameter variation detection delay versus minimizing erratic bank changes (moves, expansions or contractions). The aim of this dissertation is to exploit information available in the MMAE, in a more systematic and theoretically sound manner than

previous methods, to make good decisions for bank motion and expansion/contraction. Therefore, the use of *ad hoc* methods such as this should be avoided when possible.

3.1.5.4 On-Line Sheldon Discretization

The density algorithm provides intelligent decision making for movement, contraction and expansion of the bank. However, it relies heavily on uniform spacing of the parameter values within the newly declared parameter span of the bank. This uniform spacing is not inherent in the algorithm, and this section proposes a combination of the basic density algorithm with a discretization technique that does not rely on simple uniform spacing of the parameter values. A discretization algorithm developed by Sheldon [68, 69] provides a method to choose parameter values intelligently for an MMAE bank, given minimum and maximum values for each scalar parameter. An obvious combination is to use the density algorithm to determine the endpoints for the parameter values and Sheldon's algorithm to discretize the filter-assumed parameter values between these endpoints.

Section 2.2.3.2 introduced the basic algorithm developed by Sheldon [68] and the concept of applying a finite horizon to the iterative relationship given by

$$\Gamma_j(t_i) = \mathcal{L}\Gamma_j(t_{i-1})\mathcal{L}^T + \mathbf{G}_0\mathbf{Q}_0\mathbf{G}_0^T$$

Recall that the finite horizon is used in lieu of having steady state Kalman gains available. One case where steady state gains are not achievable is for problems with nonlinear system models leading to the use of extended Kalman filters or other approximate nonlinear filters. Additionally, if the problem is defined by a linear or linearized system model that is astable or unstable, then steady state gains are not achievable. The length of the finite horizon is motivated by the designer's physical insights into how many sample periods into the future the discretization algorithm should look when performing its vector minimization, i.e., what period of time into the future makes physical sense in the specific problem context as a period of *current* concern for performance.

The next issue is how to incorporate either the basic Sheldon algorithm or the modified Sheldon discretization (i.e., Sheldon's basic algorithm and the use of a finite horizon) with the basic density algorithm. The following method can utilize either version of the Sheldon algorithm, but emphasis is placed on the modified version since it will be applied to the example problem in Chapter 4. The Sheldon algorithm requires the designer to specify the admissible parameter region, i.e., minimum and maximum values for the elements of the parameter vector \mathbf{a} . Obviously the number of filters in the MMAE and the underlying basic system model must also be specified, but the focus here is on the range of parameter values used in the discretization process. The density algorithm provides precisely this information when it moves, contracts and expands the MMAE bank based on its decision logic. Specifically, the density algorithm will provide the endpoints of the newly declared parameter range, and either version of Sheldon's algorithm can be used to discretize the filter-assumed parameter values between these endpoints. This essentially converts Sheldon's algorithm from being used exclusively *a priori* off-line to a new on-line discretization method. There are, of course, some implementation issues associated with on-line use.

Sheldon is quick to point out that his algorithm is computationally intensive [68], and adequate minimization of the cost functionals would be cumbersome if performed at every sample time. This motivated the development of a look-up table concept which allows *a priori* application of Sheldon's discretization to a wide range of parameter regions, i.e., several choices of parameter value endpoints. The look-up table is accessed on-line to provide the best choice for the filter-assumed parameter values in a Sheldon cost minimization sense. Clearly the look-up table has a finite number of entries; so the entry with endpoints that are closest to the endpoints dictated by the density algorithm is used. The look-up table process requires minimal computation time and can be performed at every sample time.

The on-line Sheldon algorithm with a finite horizon was implemented using MATLAB [41] for the example problem given in Chapter 4. MATLAB code was developed and written to generate the required parameter sets, including a formatted look-up table that can be read into any FORTRAN routine [28]. The modified Simpson's rule presented in Appendix B was the numerical integration technique used, and the vector minimization was accomplished using MATLAB's constrained optimization techniques which are based on Sequential Quadratic Programming (SQP). Specifically, the command "*constr*" in MATLAB's Optimization Toolbox performs constrained nonlinear optimization for multivariable functions [40]. The application of the Sheldon algorithm with a finite horizon to the fixed-bank MMAE is presented in Section 4.6, and the performance attributes associated with the combination of the density algorithm with the on-line Sheldon enhancement are presented in Section 4.8.

3.2 Probability Algorithm

The second algorithm developed in the research uses the concept of parameter position estimate monitoring discussed in Section 2.2.3 in conjunction with a new probability-based discretization method. The major contribution lies in the development of the new discretization method, which is presented first, followed by the description of the probability algorithm as a stand-alone adaptive algorithm driving a moving-bank MMAE.

The fundamental concept employed by the probability-based discretization method, referred to as the PBDM, is to choose the parameter values for the elemental Kalman filters in an MMAE based on the calculation of the probability $P_{\chi}(\chi_j \leq T)$. Recall that each elemental filter assumes a value for the parameter vector, \mathbf{a}_j , and the true parameter vector is identified as \mathbf{a}_t . If any mismatching of the true system is present in the filter model ($\mathbf{a}_j \neq \mathbf{a}_t$), then χ_j is the *generalized* Chi-Squared random variable discussed throughout this dissertation and defined by the following quadratic form

of the measurement residuals (the difference between $\mathbf{z}_t(t_i)$ based on \mathbf{a}_t , and $\mathbf{H}_j(t_i)\hat{\mathbf{x}}_j(t_i^-)$ based on \mathbf{a}_j), \mathbf{r}_j , and their associated filter-computed covariance matrix, \mathbf{A}_j :

$$\chi_j = \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \quad (113)$$

The probability calculation, $P_\chi(\chi_j \leq T)$, is the probability that the *generalized* Chi-Squared variable will lie below a threshold, T . The same mathematics presented so far parallel the discussion in Section 3.1.3 under measure M_7 which alluded to calculating the above probabilities numerically for the new discretization method being presented here. Recall that χ_j provides an indication of the correctness of a filter in terms of how well the filter-assumed parameter values match the true parameter values. Specifically, a value of $\chi_j \leq m$ (the measurement dimension) indicates that the filter model is a good match to truth, whereas $\chi_j \gg m$ implies a bad match to truth. $P_\chi(\chi_j \leq T)$ is statistically stating the probability that filter j is a good match to truth for the threshold value selected. The new discretization method will exploit this probability information and use it as a basis for assigning the parameter values to the filters in the bank. The basic idea of this method will be introduced through the following example.

Consider an MMAE with five filters and the goal of accurately estimating a scalar parameter value. One approach would be to assign the filters parameter values such that the filter probabilities, $p_j(t_i)$, as calculated by the MMAE using Equation (44) on page 29 would result in the values shown in Table 2. These choices will be motivated shortly. The Chi-Squared probability value, $P_\chi(\chi_j \leq T)$, associated with each p_j value is shown in Table 2. For convenience, the shorthand notation $P_{\chi_j} = P_\chi(\chi_j \leq T)$ will be used in much of the discussion. It is important to recognize the

Table 2. Example Probability Values

Filter #	1	2	3	4	5
MMAE Probability Value, p_j	0.01	0.24	0.5	0.24	0.01
Chi-Squared Probability Value, $P_\chi(\chi_j \leq T)$	0.01	0.24	0.5	0.76	0.99

difference between the MMAE probabilities, p_j , and the probabilities, P_{χ_j} , associated with the Chi-Squared variable. These differences will be highlighted in the next paragraph following the motivation for the chosen values of the MMAE probabilities, p_j . Notice that the center filter in the bank has a relatively high probability, p_j , its neighbors have less probability but still relatively substantial amounts, and the outlying filters have extremely small p_j values. These values are determined in an *ad hoc* fashion, giving the designer significant latitude in choosing how broad or narrow to make the filter bank. The motivations for the p_j values chosen in this example parallel the thoughts presented in the density algorithm discussion of contracting the bank (see page 85). Specifically, it is desirable to give the MMAE both a foveal and peripheral view of the parameter space.

In order to strive for these desired p_j values, as calculated by the MMAE, the designer would select the P_{χ_j} values shown in Table 2. The values for P_{χ_j} may seem more important, since they are chosen by the designer and used in the PBDM, however, they are motivated by the values shown here for p_j . Both probabilities give *indications* of how well each elemental filter matches truth. However, due to the structure of the MMAE, the sum of the p_j 's will always equal one; so a filter that is actually a poor match to truth could have a value of $p_j = 0.9$ (indicating that it is a good match to truth) simply because it is a better match to truth than all the other elemental filters. In contrast, the sum of the P_{χ_j} does not equal one, in general, and if all the filters are a poor match to truth, then all of their associated P_{χ_j} values would be small. Note that $P_{\chi_j} = P_{\chi}(\chi_j \leq T)$ is a monotonic function of T , depicting the *cumulative* probability associated with χ_j taking on values *less than or equal to* T (*not* a conditional probability that χ_j takes on values in a small neighborhood *about* T). The PBDM will select parameter values for the filters such that the desired "correctness" of the filter models (in terms of how well the filter-assumed parameter values match the true parameter values) will be attained. In other words, the filter-assumed parameter values will be chosen by the PBDM such that the P_{χ_j} values shown in Table 2 (or any other values selected by the designer) will

be attained. Sections 3.2.1 and 3.2.2 present the methods used to calculate P_{χ_j} , and Section 3.2.3 explains the monotonic increase that is apparent for the P_{χ_j} values along with justification for the specific values shown here. The discussion that follows will detail the methodology and necessary calculations required to implement the PBDM.

3.2.1 Probability Calculation and Coordinate Transformation

The first tool needed for the PBDM is an analytical expression for the probability calculation given by P_{χ_j} in terms of the density function associated with the *generalized* Chi-Squared random variable. The probability that $\chi_j \leq T$ is given by the one-dimensional integral

$$P_{\chi_j} = P_{\chi}(\chi_j \leq T) = \int_0^T f_{\chi}(c) dc \quad (114)$$

where $f_{\chi}(c)$ is the *generalized* Chi-Squared density function. If the filter is a perfect match of the true system ($\mathbf{a}_j = \mathbf{a}_t$), then the density function, $f_{\chi}(c)$, is exactly the Chi-Squared density function given by Equation (93) and the probability, P_{χ_j} , can be readily calculated. However, if any mismodeling of the true system is present in the filter model ($\mathbf{a}_j \neq \mathbf{a}_t$), then the *generalized* Chi-Squared density function $f_{\chi}(c)$ is not readily known, as first mentioned in the discussion of measure M_7 . Mathematical analysis did result in the derivation of the first two moments of the *generalized* Chi-Squared random variable χ in the presence of mismodeling, and these are presented in Appendix A. Therefore, it is necessary to evaluate an m-dimensional integral based on the density function for the residuals, $f_{\mathbf{r}}(\boldsymbol{\rho})$, which *can* be expressed in the presence of mismodeling. This multi-dimensional integral is given by

$$P_{\chi}(\chi_j \leq T) = \int \cdots \int_{\mathcal{A}_+} f_{\mathbf{r}}(\boldsymbol{\rho}) d\boldsymbol{\rho} \quad (115)$$

where \mathcal{A}_r represents the appropriate region of integration in the parameter space. The functional mapping of the random variable \mathbf{r} to the random variable χ is defined by Equation (113).

The density function, $f_r(\rho)$, is fully characterized as a normally distributed random variable with mean, μ_j , and covariance matrix, \mathbf{A}_t .

$$\mathbf{r}_j : N[\mu_j, \mathbf{A}_t] \quad (116)$$

Hanlon [22] developed expressions for the residual mean vector, μ_j , in the presence of mismodeling and his results are summarized in Appendix C. Additionally, Hanlon showed that the *actual* covariance matrix for the residuals in the presence of system mismodeling is the *true* covariance matrix, \mathbf{A}_t , associated with the residuals and is computed in terms of the true system matrices as

$$\mathbf{A}_t = \mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t \quad (117)$$

where \mathbf{P}_t^- is the true error covariance matrix committed by a filter based on \mathbf{a}_t against a “real-world” based on \mathbf{a}_t (the superscript denotes the matrix prior to a measurement update), \mathbf{H}_t is the true system output matrix and \mathbf{R}_t is the true measurement noise covariance matrix. Similarly, the *filter-computed* covariance matrix is given by

$$\mathbf{A}_j = \mathbf{H}_j \mathbf{P}_j^- \mathbf{H}_j^T + \mathbf{R}_j \quad (118)$$

where \mathbf{P}_j^- is the elemental Kalman filter state estimate covariance matrix prior to a measurement update, \mathbf{H}_j is the elemental Kalman filter system output matrix and \mathbf{R}_j is the elemental Kalman filter measurement noise covariance matrix. Although Hanlon only considered mismodeling of the elemental filter model control input matrix, \mathbf{B}_{dj} , the elemental filter model output matrix, \mathbf{H}_j , and the elemental filter model state transition matrix, Φ_j , his work is easily extended to incorporate mismodeling of the elemental filter measurement noise covariance matrix, \mathbf{R}_j , and the elemental filter dynamics driving noise covariance, \mathbf{Q}_{dj} . Specifically, since Equation (117) is independent of \mathbf{R}_j and \mathbf{Q}_{dj} , the *true* covariance matrix, \mathbf{A}_t , associated with the residuals is still given by Equation (117). The mean of the residual in the presence of \mathbf{R}_j and \mathbf{Q}_{dj} mismodeling is presented in Appendix

C as being zero-mean. Therefore, a fully characterized density function, $f_{\mathbf{r}}(\boldsymbol{\rho})$, is available for use in the integration given by Equation (115).

It is possible to simplify this integration and desirable to formulate a closed form solution that would avoid the use of numerical integration. Unfortunately, a closed form solution was not found because of the subtle yet significant difference in the covariance matrix, \mathbf{A}_j , that appears the quadratic

$$\chi_j = \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$$

and the *true* covariance matrix associated with the residuals, \mathbf{A}_t , i.e., $\mathbf{A}_j \neq \mathbf{A}_t$. Maybeck [44] utilized a coordinate translation and transformation to find a closed form for a similar integration, given that the covariance matrix used in the quadratic is \mathbf{A}_t . Specifically, problem 6.10 of [44] assumed that the quadratic would be of the form

$$\chi_j = \mathbf{r}_j^T \mathbf{A}_t^{-1} \mathbf{r}_j$$

Despite the inability to apply Maybeck's solution directly, considerable insight was gained into the use of the coordinate transformation to simplify the numerical integration that would now be necessary. The density function, $f_{\mathbf{r}}(\boldsymbol{\rho})$, of the m-dimensional residual can be visualized by means of a hyperellipsoid contour map, with a two-dimensional example illustrated in Figure 32. Integration over the shaded (hyper)elliptical region, $\mathcal{A}_{\mathbf{r}}$, would be messy in terms of the limits of integration. However, this can be simplified if the region is transformed to the shaded (hyper)spherical region, $\mathcal{A}_{\mathbf{r}'}$, associated with the density function, $f_{\mathbf{r}'}$, as shown in Figure 33. The development presented here will show that the transformed limit of integration is a radius of integration equal to the square root of the threshold, \sqrt{T} . The examples shown below assume the residuals are zero-mean for illustrative purposes when, in general, one must consider the non-zero mean case shown in Figure 34 (recall that the residuals *are* zero-mean for parameters affecting \mathbf{R} and \mathbf{Q}_d , but not for parameters affecting $\boldsymbol{\Phi}$, \mathbf{B}_d , or \mathbf{H}).

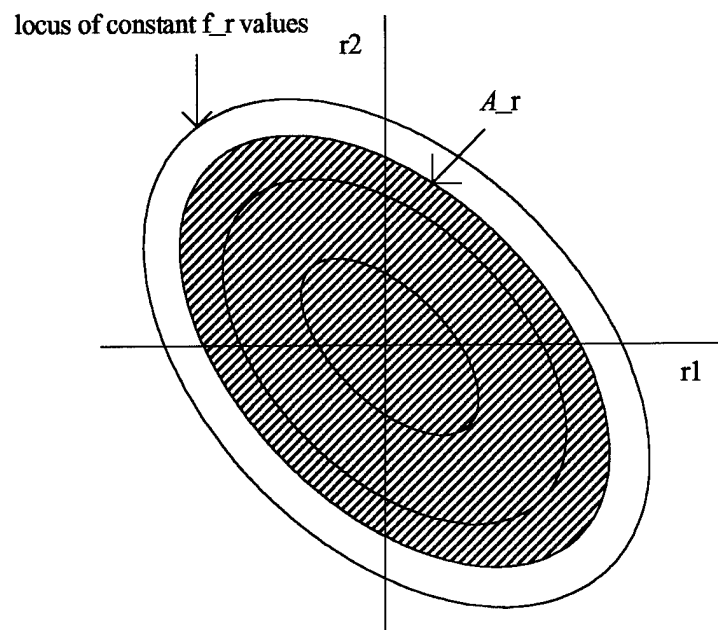


Figure 32. Density Function Contour Map (Hyper)ellipsoids for Two-Dimensional Residuals

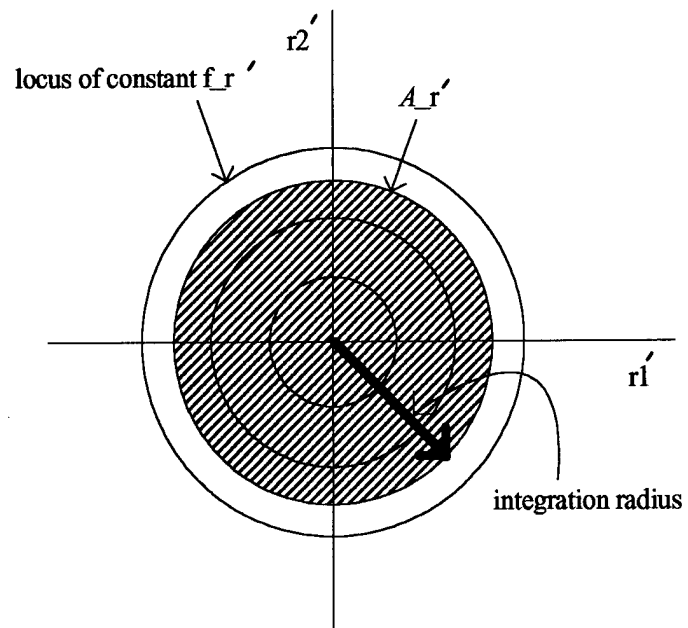


Figure 33. *Transformed* Density Function Contour Map (Hyper)spheroids for Two-Dimensional Residuals

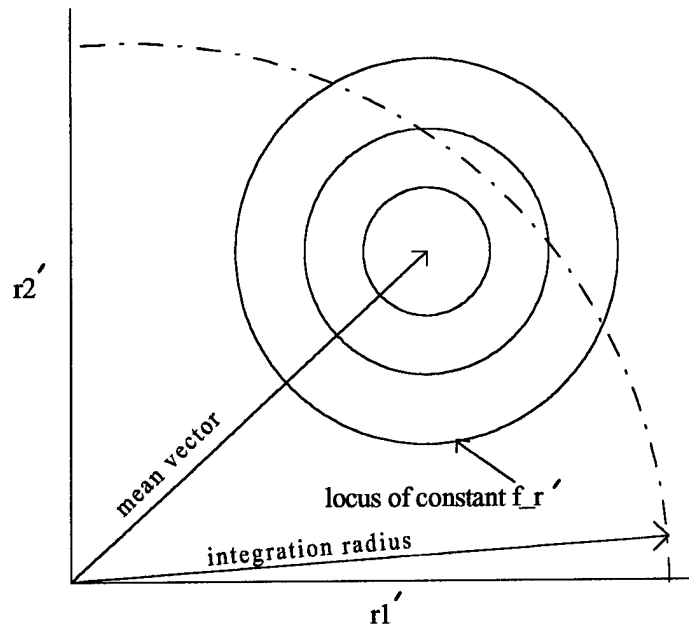


Figure 34. *Transformed* Density Function Contour Map (Hyper)spheroids for Two-Dimensional Residuals with a Non-Zero Mean

Apply the following *linear* transformation to the residuals

$$\mathbf{r}'_j = \sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \mathbf{r}_j \quad (119)$$

where a Cholesky square root of \mathbf{A}_j^{-1} is used such that

$$\mathbf{A}_j^{-1} = \left(\sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \right)^T \left(\sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \right) \quad (120)$$

This *linear* transformation on the Gaussian random variable maintains its Gaussian nature as \mathbf{r}'_j : $N[\boldsymbol{\mu}'_j, \mathbf{A}'_t]$ with the transformed mean vector and covariance matrix given by

$$\boldsymbol{\mu}'_j = \sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \boldsymbol{\mu}_j \quad (121)$$

$$\mathbf{A}'_t = \left(\sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \right) \mathbf{A}_t \left(\sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \right)^T \quad (122)$$

and the *generalized* Chi-Squared variable given by

$$\chi_j = \mathbf{r}'_j{}^T \mathbf{r}'_j \quad (123)$$

This is quickly verified by first establishing that the quadratic used to calculate χ_j is unaffected:

$$\begin{aligned} \chi_j &= \mathbf{r}'_j{}^T \mathbf{r}'_j = \left(\sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \mathbf{r}_j \right)^T \left(\sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \mathbf{r}_j \right) = \mathbf{r}_j^T \left(\sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \right)^T \left(\sqrt[{\mathcal{C}}]{\mathbf{A}_j^{-1}} \right) \mathbf{r}_j \\ &= \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \end{aligned} \quad (124)$$

Next the mean of the residual is given by

$$\mu'_j = E\{\mathbf{r}'_j\} = E\left\{\sqrt[m]{\mathbf{A}_j^{-1}}\mathbf{r}_j\right\} = \sqrt[m]{\mathbf{A}_j^{-1}}E\{\mathbf{r}_j\} = \sqrt[m]{\mathbf{A}_j^{-1}}\mu_j \quad (125)$$

Finally, the covariance matrix is given by

$$\begin{aligned} \mathbf{A}'_t &= E\left\{(\mathbf{r}'_j - \mu'_j)(\mathbf{r}'_j - \mu'_j)^T\right\} = E\{\mathbf{r}'_j\mathbf{r}'_j^T\} - E\{\mathbf{r}'_j\mu_j'^T\} - E\{\mu_j'\mathbf{r}'_j^T\} + E\{\mu_j'\mu_j'^T\} \\ &= E\left\{\left(\sqrt[m]{\mathbf{A}_j^{-1}}\mathbf{r}_j\right)\left(\sqrt[m]{\mathbf{A}_j^{-1}}\mathbf{r}_j\right)^T\right\} - \mu_j'\mu_j'^T - \mu_j'\mu_j'^T + \mu_j'\mu_j'^T \\ &= \left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)E\{\mathbf{r}_j\mathbf{r}_j^T\}\left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)^T - \sqrt[m]{\mathbf{A}_j^{-1}}\mu_j\left(\sqrt[m]{\mathbf{A}_j^{-1}}\mu_j\right)^T \\ &= \left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)E\{\mathbf{r}_j\mathbf{r}_j^T\}\left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)^T - \left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)\mu_j\mu_j^T\left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)^T \\ &= \left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)(E\{\mathbf{r}_j\mathbf{r}_j^T\} - \mu_j\mu_j^T)\left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)^T = \left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)\mathbf{A}_t\left(\sqrt[m]{\mathbf{A}_j^{-1}}\right)^T \end{aligned} \quad (126)$$

This leads to the transformation of Equation (115) into the m-dimensional integral

$$P_X(\chi_j \leq T) = \int \cdots \int_{\mathcal{A}_{\mathbf{r}'}} f_{\mathbf{r}'}(\rho') d\rho' \quad (127)$$

which is now based on the transformed density function $f_{\mathbf{r}'}(\rho')$. Finally, before delving into the numerical integration technique used to evaluate Equation (127), the radius of integration shown in Figure 33 must be identified. The integration has been simplified from hyperellipsoids to hyperspheroids, and Equation (123) can be recognized as the equation for a hypersphere; so the upper limit of integration is simply some radius of this hypersphere. Furthermore, the original limit of integration for $P_X(\chi_j \leq T)$ in Equation (114) is the threshold, T , and from the relationship of χ_j to \mathbf{r}' given by Equation (123), the radius of integration is recognized as the square root of the threshold,

\sqrt{T} . The discretization method discussed in Section 3.2.3 will provide one method for calculating T .

3.2.2 Multi-Dimensional Numerical Integration

The previous section stated the need to perform numerical integration for the evaluation of Equation (127). Most integration techniques presented in the open literature address numerical integration up to three dimensions. However, an m -dimensional numerical integration routine is required here, where m is the number of measurements being fed to the MMAE. The following integration method was provided by Oxley [61] and Maybeck [44] and simply extends the often utilized technique of transforming the coordinate system from Cartesian to polar coordinates prior to performing the numerical integration.

Consider the m -dimensional integration given by

$$\int \cdots \int f(r_1, r_2, \dots, r_m) dr_1 dr_2 \dots dr_m \quad (128)$$

where $f(r_1, r_2, \dots, r_m)$ is the m -dimensional function that is being integrated. Transform from Cartesian to hyperspherical coordinates via the following equations

$$\begin{aligned} r_1 &= \tau \cos(\phi) \\ r_2 &= \tau \sin(\phi) \cos(\theta_1) \\ r_3 &= \tau \sin(\phi) \sin(\theta_1) \cos(\theta_2) \\ &\vdots \\ r_{m-1} &= \tau \sin(\phi) \sin(\theta_1) \sin(\theta_2) \dots \sin(\theta_{m-3}) \cos(\theta_{m-2}) \\ r_m &= \tau \sin(\phi) \sin(\theta_1) \sin(\theta_2) \dots \sin(\theta_{m-2}) \end{aligned}$$

where

τ = radius of the hyperspheroids

for ϕ and θ_k within the limits

$$0 \leq \phi \leq 2\pi \quad -\frac{\pi}{2} \leq \theta_k \leq \frac{\pi}{2} \quad \forall k = 1, 2 \dots m-2$$

and the differential hypervolume dV is given by

$$dV = dr_1 dr_2 \dots dr_m = |\mathbf{J}| d\tau d\phi d\theta_1 \dots d\theta_{m-2} \quad (129)$$

where \mathbf{J} is the Jacobian matrix given by

$$\mathbf{J} = \begin{bmatrix} dr_1/d\tau & dr_1/d\phi & \dots & dr_1/d\theta_{m-2} \\ dr_2/d\tau & dr_2/d\phi & \dots & dr_2/d\theta_{m-2} \\ \vdots & \vdots & & \vdots \\ dr_m/d\tau & dr_m/d\phi & \dots & dr_m/d\theta_{m-2} \end{bmatrix} \quad (130)$$

In fact, the determinant of the Jacobian is given in the new coordinates as

$$|\mathbf{J}| = \text{Abs}[\tau^{m-1} \sin^{m-2}(\phi) \sin^{m-3}(\theta_1) \sin^{m-4}(\theta_2) \dots \sin(\theta_{m-3})] \quad (131)$$

where “Abs” refers to the absolute value. The user defines the spacing between discrete evaluation points by setting $d\tau$, $d\phi$ and $d\theta_1 \dots d\theta_{m-2}$. Now the numerical integration is performed by summing up the product of the function and the differential hypervolume evaluated at the user defined discrete points $\mathbf{r} = [r_1, r_2, \dots r_m]^T$ as shown in Equation (132):

$$\sum_{\tau} \sum_{\phi} \sum_{\theta_1} \dots \sum_{\theta_{m-2}} f(r_1, r_2, \dots r_m) dV \quad (132)$$

The number of discrete values used for the evaluation points will dictate the accuracy and computer processing time required to perform the integration.

In summary, evaluation of Equation (127) is performed numerically using Equation (132) where

τ = is the radius of integration that varies from 0 to \sqrt{T}

ϕ and θ_k have set ranges previously defined

dV is calculated using Equation (129)

$d\tau, d\phi$ and $d\theta_k$ are user specified

and $f(r_1, r_2, \dots, r_m)$ is given by a normal density function with mean, μ'_j , and covariance, A'_t (see Equations (121) and (122))

$$f(\mathbf{r}) = f(r_1, r_2, \dots, r_m) = \frac{1}{(2\pi)^{m/2} |A'_t|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{r} - \mu'_j)^T A'^{-1}_t (\mathbf{r} - \mu'_j) \right\}$$

3.2.3 Discretization Method

The discretization method discussed in this section is one of many possible which utilize the tools presented in the previous two sections. Other concepts including methods to move, contract, and expand the MMAE bank based on the numerical evaluation of P_{x_j} could be explored. The method of choosing the parameter values for the elemental Kalman filters based on the calculation of the probability, P_{x_j} , allows the algorithm to be adaptive and replace previous *ad hoc* methods such as uniform spacing of the parameter values. Additionally, it takes into account the system's sensitivities to the parameters being estimated, which are often not precisely known to a designer who is choosing the filter-assumed parameter values in an *ad hoc* manner.

Return to the five-filter example with the arbitrarily chosen probability values, P_{x_j} , shown in Table 2 on page 110. In order to integrate these probabilities numerically with Equation (132), it is necessary to have the mean vector and covariance matrix for the residuals based on both a truth model and a filter model. Since the parameter value for the truth model is not exactly known (otherwise there would be no estimation needed), it is necessary to assume values for the truth

model based on the best guess given by the MMAE. Specifically, assume the true parameter values are equal to the components of $\hat{\mathbf{a}}_{\text{MMAE}}(t_i)$. This best guess is available on-line, and by design the filter in the center of the bank will assume its value.

Given this assumption, the threshold, T , can be determined exactly for *any* choice of P_{χ_t} , where the subscript t indicates that the filter-assumed parameter values are assumed to match truth. Recall that, if the filter is a perfect match of the true system, then the density function $f_{\chi}(c)$ is exactly the Chi-Squared density function given by Equation (93) and the probability can be readily calculated. Inversely, the threshold can be found using the MATHCAD function " $T = qchisq(P_{\chi_t}, m)$ " which is a function of both the desired probability for the filter assumed as truth and the dimension of the residuals [39]. Note that T is not a function of $\hat{\mathbf{a}}_{\text{MMAE}}(t_i)$; so once P_{χ_t} and m are set, T will be known for all values of $\hat{\mathbf{a}}_{\text{MMAE}}(t_i)$.

The remaining four filters will be assigned parameter values above and below $\hat{\mathbf{a}}_{\text{MMAE}}(t_i)$ such that their P_{χ_j} values agree with those shown in Table 2 for the now known threshold value, T . This raises the issue of how to select parameter values for the remaining four filters without simply running multiple guesses through the numerical integrator and hoping that four of them generate the desired P_{χ_j} values. An automated search algorithm will be presented shortly to overcome this problem.

A second issue is how to select the P_{χ_j} values shown in Table 2 (page 110) in order to strive for the desired p_j values. Recall that the p_j values were chosen to meet the design goal of giving the MMAE both a foveal and peripheral view of the parameter space. Engineering intuition supports the proposition that selecting parameter values for filter 3 in this five-filter example such that $P_{\chi_3} = 0.5$ (i.e., this filter will be a good match to truth with a probability of 0.5) is equivalent to selecting its parameter values such that $p_3 = 0.5$ (i.e., the MMAE will assign a probability weight of 0.5 for this filter). This holds for all filters with $j < \arg[p_{\max}]$, i.e., filters 1 and 2 for this example. Therefore,

$P_{\chi_j} = p_j$ for $j = 1, 2, \dots, \arg[p_{\max}]$ as shown in Table 2. Additional insights are required prior to realizing that $P_{\chi_j} = 1 - p_j$ for $j > \arg[p_{\max}]$, i.e., filters 4 and 5 for this example. The values in Table 2 of $P_{\chi_4} = 0.76$ and $P_{\chi_5} = 0.99$ may seem counterintuitive, since these filters will be assigned parameter values that are not designed to match truth as well as filter 3, but their probability, P_{χ_j} , is selected greater than P_{χ_3} , implying that they are a better match to truth.

This phenomenon is best understood (but not limited to) the case in which the parameter value being estimated is the measurement noise covariance \mathbf{R} , and empirical studies provided validation of this concept for this case. The reason is clear when looking at Equations (117) and (118) on page 113, which are dependent on the choices of the measurement noise covariances \mathbf{R}_t and \mathbf{R}_j respectively. Given that \mathbf{R}_t is selected as $\hat{\mathbf{a}}_{\text{MMAE}}$ (the truth model is based on the best guess from the MMAE), when \mathbf{R}_j takes on values greater than \mathbf{R}_t , the eigenvalues of the associated residual covariance matrix, \mathbf{A}_j , will increase. This directly effects \mathbf{A}'_t through Equation (122), resulting in smaller eigenvalues for \mathbf{A}'_t . The discussion in Appendix C states that the residuals will be zero-mean when mismodeling the measurement noise covariance. This results in numerically integrating density functions similar to the two-dimensional example shown in Figure 33. Given that the eigenvalues of \mathbf{A}'_t have been reduced, the numerical integration and thus probability value, P_{χ_j} , for a given integration radius will increase. In other words, given that the eigenvalues of \mathbf{A}'_t have been reduced, the density function has become highly peaked with more of the probability density residing about its zero mean. Therefore, the numerically integrated probabilities, P_{χ_4} and P_{χ_5} , for filters 4 and 5 will be greater than the probability, P_{χ_3} , for filter 3. It is anticipated that mismodeling in any of the other system matrices (\mathbf{Q}_{dj} , Φ_j , \mathbf{H}_j , and \mathbf{B}_{dj}), will result in this same attribute, and this is a topic for future research. To account for this phenomenon, choose the parameter values such that the *probability variation* between the center filter and its neighbors is used in conjunction with the probability, P_{χ_j} , calculated through Equation (132).

Continue with the example problem of five filters and focus on the three inner filters (filter numbers 2 - 4). The above phenomenon is illustrated in Figure 35, where the probabilities, P_{χ_j} , are shown to grow faster towards one for filters with higher assumed values of R_j . The *probability*

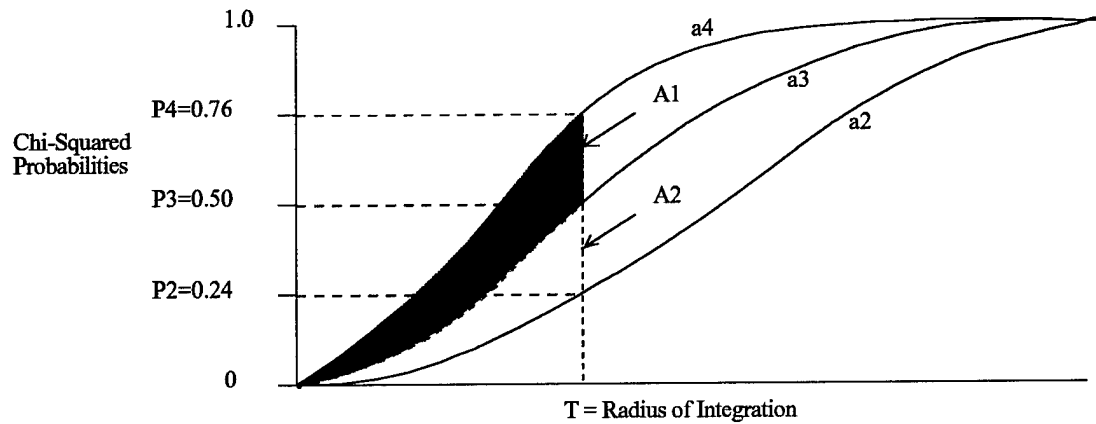


Figure 35. Example Integration Curves with Equal Probability Variation

variation between filters 3 and 4 is shown as A_1 and between filters 2 and 3 as A_2 . Recall the design choices for the probabilities of filters 2 and 3 as $P_{\chi_2} = 0.24$ and $P_{\chi_3} = 0.5$. The probability variations are then given by

$$A_1 = P_{\chi_4} - P_{\chi_3}$$

$$A_2 = P_{\chi_3} - P_{\chi_2}$$

One approach is to require equal probability variations such that

$$A_1 = A_2$$

$$\Rightarrow P_{\chi_4} - P_{\chi_3} = P_{\chi_3} - P_{\chi_2}$$

$$\Rightarrow P_{\chi_4} = 2P_{\chi_3} - P_{\chi_2}$$

and for this example $P_{\chi_4} = 0.76$. Therefore, when choosing a parameter value for filter 4, the goal is to find a numerically integrated probability of 0.76. Similarly, for filter 5, $P_{\chi_6} = 0.99$. Notice that, in general, $P_{\chi_j} = 1 - p_j$ for $j > \arg[p_{\max}]$.

The discretization method presented in this section provides a more systematic and more theoretically substantiated means of choosing the filter-assumed parameter values, \mathbf{a}_j , than the *ad hoc* methods used to date. Specifically, the designer's primary concern is to select desired probability p_j values, leading to the associated P_{χ_j} values such as those in Table 2, then apply the PBDM to determine the appropriate \mathbf{a}_j values.

3.2.3.1 Parameter Value Search Routine

As mentioned in Section 3.2.3, an automated search routine is presented next which greatly simplifies the process of finding parameter values for the filters to meet the desired probability goals. This routine has only been developed for the special case of zero-mean residuals in the presence of mismodeling and a *scalar* parameter. Recall that Appendix C identified the special case of zero-mean residuals as associated with \mathbf{Q}_d (dynamics noise covariance) and \mathbf{R} (measurement noise covariance) mismodeling. It is believed that a similar search routine could be developed for the nonzero-mean case and some ideas will be presented along these lines.

The search routine is based on a measure of the eigenvalues of the transformed residual covariance matrix \mathbf{A}'_t . Recall the above discussion explaining how the eigenvalues of \mathbf{A}'_t varied with the choice of the parameter \mathbf{R} resulting in the various values for the probabilities $P_{\chi}(\chi_j \leq T)$. It is possible to use a measure of the eigenvalues of \mathbf{A}'_t to predict P_{χ_j} without evaluating this probability numerically. Specifically, the product of the eigenvalues of \mathbf{A}'_t was found to predict P_{χ_j} consistently:

$$M_{eig} = \prod_{i=1}^m \lambda_i \quad (133)$$

where

λ_i = the eigenvalues of A'_t

This makes sense since the product of the eigenvalues of a matrix is directly related to the volume of the hyperellipsoid defined by that matrix [73]. The principle axes of this hyperellipsoid are represented by the eigenvectors of the matrix. The flowchart of the search routine based on the measure shown in Equation (133) is illustrated in Figure 36. Again, each major event or decision is labeled with a number to help the reader relate the forthcoming discussion with the flowchart.

The basic process used by the search routine is repeated for each elemental filter. First, assume a parameter value for the elemental filter, i.e., choose a “guess” value. Minimum and maximum guess values will be discussed presently, but for now, recognize that the search routine will begin at either a minimum or maximum guess value. Calculate the measure, M_{eig} , associated with this guess value and determine if the search criterion defined below is met. If so, the search is complete and repeat for the other elemental filters. If not, make another guess for the parameter value based on a guess step size described below and check against the criteria. Continue this process via a loop until the criterion is met or a set number of guesses have been tried.

1. *Set Search Criteria*: The search criteria include:

- \hat{a}_{MMAE} = scalar true parameter value (current best guess)
- M_{eig}^d = desired eigenvalue measure
- a_l = minimum scalar guess value, see block 3 “Initialize Search ”
- a_h = maximum scalar guess value, see block 3 “Initialize Search ”
- N_g = number of guesses of parameter values (for each elemental filter) for the search

The value of \hat{a}_{MMAE} is given by the MMAE. The desired eigenvalue measure, M_{eig}^d , must be empirically determined for each filter. Recall that one filter (typically the center filter) takes on the value of \hat{a}_{MMAE} and is assumed to match truth, so there is no need to search for this parameter value, and thus there is no need to find M_{eig}^d for this filter. A manual search must be done *once* for

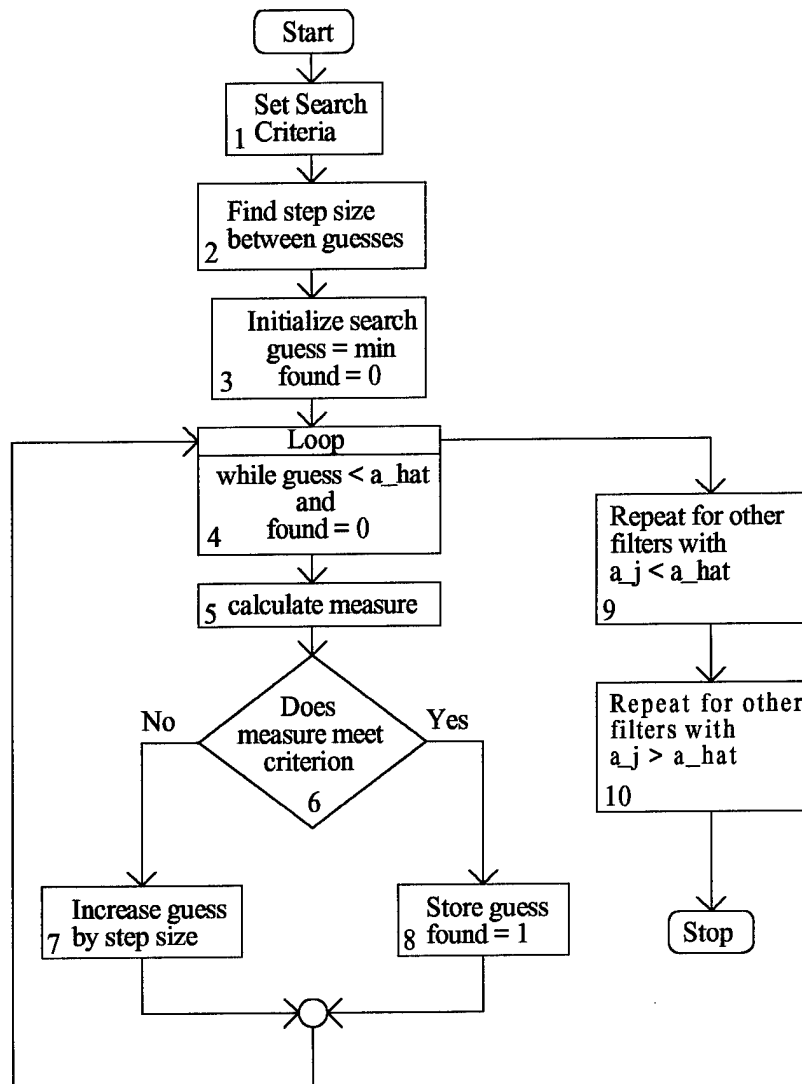


Figure 36. Parameter Value Search Routine

each of the $J-1$ values of M_{eig}^d , as explained below. The phrase manual search refers to arbitrarily selecting a value (a guess) for the parameter a_j , then performing the numerical integration to determine if the resulting $P_x(\chi_j \leq T)$ is equal to the desired probability value in Table 2. Once a parameter value is found that produces the appropriate probability value, P_{x_j} , the eigenvalues of A'_t associated with this parameter value are used to calculate M_{eig}^d via Equation (133). This process is repeated to determine each of the $J-1$ values of M_{eig}^d . However, after M_{eig}^d is determined *once* for any single value of \hat{a}_{MMAE} , this search routine will replace the manual method for all other values of \hat{a}_{MMAE} . Section 3.2.3.2 will discuss implementation issues and explain the need to perform the automated search many times to create a look-up table for on-line use. It is recommended to perform the manual search with a value of \hat{a}_{MMAE} that is approximately centered in the parameter space. More importantly, avoid using the edges of the parameter space for this manual search, since there will be no room to place parameters in one direction.

Intelligent choices for a_l and a_h will be discussed in block 3, “*Initialize Search.*” The number of guesses for each elemental filter is arbitrary and often less than 100 for most problems. Large values of N_g will slow down the search routine and rarely provide noticeable improvements in the final probability calculations. This is largely dependent on the system’s sensitivity to the parameter in question, but in most cases the numerical precision of the integration is not sufficient to delineate between a search with 50 guesses and a search with 80 guesses.

2. *Find Step Size*: The step size between guesses for each elemental filter is calculated for the scalar parameter as

$$S_j = \begin{cases} \frac{\hat{a}_{MMAE} - a_l}{N_g} & \text{for filters assuming } a_j < \hat{a}_{MMAE} \\ \frac{a_h - \hat{a}_{MMAE}}{N_g} & \text{for filters assuming } a_j > \hat{a}_{MMAE} \end{cases} \quad (134)$$

where

$$a_j = \text{assumed parameter value for filter } j$$

Notice that S_j is dependent on whether the elemental filter's parameter value is assumed to lie above or below \hat{a}_{MMAE} .

3. *Initialize Search* : Start the guess value at the minimum guess value and set a flag used to indicate when the criterion is met ($\text{found} = 0$). The search time for the routine is heavily dependent on the chosen minimum and maximum guess values. An obvious choice might be the minimum and maximum allowable parameter values, however, these would be poor choices in most cases. A better approach is to let

$$\begin{aligned} a_l &= \delta_l \cdot \hat{a}_{\text{MMAE}} \\ a_h &= \delta_h \cdot \hat{a}_{\text{MMAE}} \end{aligned}$$

where δ_l and δ_h represent arbitrary fractions of \hat{a}_{MMAE} . Typical values for a five-filter problem are shown in Table 3. This approach reduces the parameter space used in the search and significantly decreases computer processing time.

Table 3. Example Fraction Values

Filter #	1	2	3	4	5
Fraction Value	0.05	0.4	-	1.3	1.5

4. *While Loop* : The flowchart symbol used here has the following interpretation. The loop is entered from the top, i.e. after block 3 is complete. Blocks 5 – 8 are repeatedly processed until either the measure criterion is met or the guess value exceeds \hat{a}_{MMAE} . The loop is exited to the right to begin processing of block 9.

5. *Calculate Measure* : The eigenvalue measure, M_{eig} , is found via Equation (133), which requires calculation of Equations (117), (118) and (122). One of the primary motivations for using this search routine is the ability to avoid the computationally intensive numerical integration required to calculate $P_x(\chi_j \leq T)$. Equations (117), (118), (122) and (133) require much less computer processing time than the numerical integration.

6. *Check If Criterion is Met*: Compare M_{eig} to M_{eig}^d and set the flag (found = 1) when the measure criterion is met. Notice that, since the search is increasing (decreasing) the guess value toward \hat{a}_{MMAE} as explained in block 7, the criterion is met if M_{eig} equals or passes by M_{eig}^d . For the cases in which $a_j < \hat{a}_{MMAE}$, M_{eig} will initially be greater than M_{eig}^d . As the search continues (the guess value increases), M_{eig} will decrease and eventually equal or pass M_{eig}^d . Recall that a_j is initially much less than \hat{a}_{MMAE} , resulting in large eigenvalues for A'_t and thus a large value for M_{eig} . As the guess value increases, a_j gets closer to \hat{a}_{MMAE} , resulting in smaller eigenvalues for A'_t and thus a smaller value for M_{eig} . Similarly for $a_j > \hat{a}_{MMAE}$, but with M_{eig} increasing from a small value as the guess value decreases. The specific logic is given by

IF $(M_{eig} \leq M_{eig}^d)$ THEN (found = 1) for $a_j < \hat{a}_{MMAE}$

IF $(M_{eig} \geq M_{eig}^d)$ THEN (found = 1) for $a_j > \hat{a}_{MMAE}$

7. *Increase Guess by Step Size*: If the criterion is *not* met, then the guess value is increased by the step size S_j for filters assuming $a_j < \hat{a}_{MMAE}$ or decreased by S_j for filters assuming $a_j > \hat{a}_{MMAE}$.

8. *Store Guess*: If the criterion is met, then the guess value is stored as the chosen parameter value for this filter and the found flag is set to 1, allowing the search to terminate.

9. & 10. *Repeat for Other Filters*: Simply repeat the process for the remaining filters. Again, the search is performed by increasing the guess value when $a_j < \hat{a}_{MMAE}$ and decreasing the guess value for $a_j > \hat{a}_{MMAE}$.

It is important to validate that the search routine has found parameter values which result in the designer-chosen values for P_{χ_j} . This is done by numerically integrating via Equation (132) for each parameter value. Given that the validation is successful, the discretization process is complete for the chosen value of \hat{a}_{MMAE} .

Recall that this routine has only been developed for the special cases of zero-mean residuals in the presence of mismodeling and a scalar parameter. This allowed the search routine to focus on the eigenvalue measure given by Equation (133) since all the densities (matched or mismatched to truth) are centered at the origin. A similar search routine could be developed that incorporates a second degree of freedom, namely the magnitude of the *transformed* residual mean vector:

$$M_{\mu'_j} = \left[\sum_{i=1}^m (\mu'_{ij})^2 \right]^{1/2} \quad (135)$$

where

$$\mu'_j = [\mu'_{1j}, \mu'_{2j}, \dots, \mu'_{mj}]$$

For a given value of T , a large value of $M_{\mu'}$ would lead to smaller numerical probabilities, $P_x(\chi_j \leq T)$, than a small value of $M_{\mu'}$. Simply look at Figure 34 on page 117 and realize that a large residual mean vector magnitude will shift the density's region of high probability outside the integration region. Other concepts include monitoring the direction of the mean vector. This direction is *not* important once the density is transformed to the "primed" coordinates and the hyperellipsoids are transformed to hyperspheroids, i.e., the transformation from Figure 32 to Figure 33. Similarly, Figure 34 illustrates that changing the direction of the mean vector will not affect the integrated value of $P_x(\chi_j \leq T)$, i.e., moving the (hyper)spheroids from Cartesian quadrant one to any other quadrant will *not* change $P_x(\chi_j \leq T)$ unless the *magnitude* of the mean vector changes. The exact methodology used to incorporate $M_{\mu'}$ and the direction of the mean vector is a subject for future research along with the case in which the parameter vector is *not* a scalar.

3.2.3.2 Implementation Issues

Two options are available for real-time implementation of the discretization method which utilizes the numerical integration calculation and the parameter value search routine. First, at each

sample time and for the current value of \hat{a}_{MMAE} , the search routine could be used to generate candidate parameter values for the filters. This would be followed by validating that the candidate parameter values produce the design choices for $P_{\chi}(\chi_j \leq T)$ via numerical integration. This approach is extremely computer-intensive since the numerical integration can require considerable processing time for accurate results with problems having multiple measurement sources, i.e., $m > 1$. This motivates a second approach which performs the numerical integration off-line. The search routine would also be performed off-line for several discrete values of \hat{a}_{MMAE} which range over the span of the admissible parameter space. The filter-assumed parameter values determined for each discrete value of \hat{a}_{MMAE} would be stored in a look-up table and referenced by that value of \hat{a}_{MMAE} . By design, a reference value of \hat{a}_{MMAE} will be used as the parameter value for the filter residing in the center of the bank. This look-up table will be available in real-time and the table entry closest to the current value of \hat{a}_{MMAE} will be used at each sample time.

A critical assumption must be identified when applying this discretization method, regardless of which option just described is being used. Recall that the search method considers multiple guess values for the parameter values, and for each guess the residual covariance matrix must be calculated. Given that the gains and covariances are chosen to be computed on-line, a Kalman filter would be required for every possible guess value being considered. Without the use of precomputable gains and covariance matrices, a potentially infinite number of Kalman filters being run in real-time would be needed, which is clearly not practical. In order to calculate the true and filter-computed residual covariance matrices given by Equations (117) and (118), it is easier to assume that precomputable Kalman filter gains and covariances are being used (easier in the sense that the gains and covariances need not be calculated on-line).

3.2.4 Algorithm Description

The probability-based discretization method (PBDM) is now used in conjunction with parameter position estimation monitoring to form a new algorithm titled the “Probability Algorithm”. The probability algorithm executes a two-step process. First, find the new center of the bank using parameter position estimation monitoring as introduced on page 40. This first step acts as a bank movement mechanism, which if applied at *every* sample period, could induce continual transients in the state estimates. This motivates using the *ad hoc* technique of decision delays discussed in Section 3.1.5.3. Second, apply the PBDM to determine the parameter values for the remaining filters in the bank. Notice the absence of any explicit contraction or expansion methods. This algorithm simply keeps the bank centered on the best estimate for the parameters, $\hat{\mathbf{a}}_{\text{MMAE}}$ (as closely as possible in view of the discrete values of $\hat{\mathbf{a}}_{\text{MMAE}}$ stored in the look-up table), then relies on the new discretization method to determine the span of the parameter values. The same performance enhancement discussed in Section 3.1.5.2 for initialization of newly declared filters is implemented for the probability algorithm with implementation issues discussed in Chapter 4. The probability algorithm just described provides only one methodology for utilizing the PBDM bank-sizing algorithm, and other moving-bank mechanisms could be used in conjunction with the PBDM. In fact, Section 3.3 presents the idea of combining the moving-bank density algorithm with the PBDM algorithm.

3.2.5 Dead Zone for Moves

Applying a dead zone similar to the method discussed in Section 3.1.5.1 can prevent unwanted bank movements. Given that a true parameter changed, the bank would move toward and eventually encompass the true parameter within the bank’s parameter span. Once the true parameter was encompassed, an additional move might be needed to center the bank on the true parameter value. However, in many cases the algorithm induced multiple moves that would overshoot the true para-

meter and even oscillate about this true value with multiple moves to the left and right. The result was erratic parameter estimates that degraded performance. The solution was to implement a dead zone that precluded the bank from making multiple small movements. The size of the move being attempted would have to be large enough (implying that it is really needed), or the move would not be allowed. The rather simple logic is given by:

IF (Attempted Move Size < Minimum Move Size) THEN (Do Not Make Move)

The minimum move size or dead zone is empirically determined with the following guidelines. Consider the typical span of the parameter values stored in the look-up table. Specifically, observe the distances between the stored values for the bank center and their associated endpoints. These distances will indicate the amount of true parameter variation that can be tolerated without a large risk of having the true parameters move outside the current parameter span of the MMAE. A reasonable first choice for the minimum move size is one-third the average distance from the bank centers to their endpoints. This one-third factor was determined empirically through simulations. Clearly, too large of a minimum move size will cause the algorithm to become sluggish in reacting to parameter variations while too small a minimum move size will make the dead zone method ineffective, leading to nonsystematic moves in the parameter space.

3.3 Combined Density and PBDM Algorithm

Recall the discussion in Section 3.1.5.4 which stated that the density algorithm provides intelligent decision making for movement, contraction and expansion of the bank. However, it relies heavily on uniform spacing of the parameter values within any newly declared parameter span of the bank. The probability based discretization method provides a process to choose parameter values intelligently for the remaining filters in an MMAE bank, given the center parameter value for the bank. An obvious combination is to use the density algorithm to determine the center parameter

value for the bank and the PBDM to discretize the filter-assumed parameter values about this center parameter value.

Full descriptions of the decisions made by the density algorithm are given in Section 3.1.4. When making a soft move, expansion or contraction, the bank is centered on the filter-assumed parameter value having the maximum probability. This provides the new center parameter value, a_{center}^{new} , which is then used to reference the look-up table created by the PBDM. For the remaining decisions (medium and hard moves), the density algorithm only provides new parameter endpoints. One logical choice for the new center parameter value is found by taking the midpoint between the newly determined parameter endpoints and given by

$$a_{center}^{new} = \frac{a_1^{new} + a_j^{new}}{2}$$

This relatively simple approach is used since there is no motivation for choosing any other point between the new parameter endpoints. Again, a_{center}^{new} is then used to reference the look-up table created by the PBDM. The same performance enhancements discussed in Section 3.1.5 can be applied, with the exception of the on-line Sheldon discretization since the PBDM is already performing the discretization.

3.4 Chapter Summary

This chapter presented the analysis that led to the development of new algorithms for a moving-bank MMAE. The underlying concepts were described to assist the reader in understanding why certain measures were chosen for the density algorithm's decision making process. Additionally, a probability-based method was introduced as a new means of discretizing the filter-assumed parameter values in the MMAE bank. Future research could be based on these newly developed concepts, and the algorithms used in this research provide a few well-developed methods for parameter estimation.

Chapter 4 - Simulation Performance

4.1 Overview

This chapter first introduces the simulation techniques and computer software used to apply the concepts developed in previous chapters. The motivating example of parameter estimation for fault tolerant aircraft precision landing is simulated. It is not the focus of this research to provide an in-depth sensitivity analysis of the algorithms to all potential real-world scenarios that could apply to aircraft precision landing. Rather, a single failure mode representing interference/jamming of the GPS receiver is simulated for proof of concept and validation of the algorithms. An overall description of the integrated system is given, followed by detailed state and measurement models for each of the navigation subsystems used. The implementation details and performance of several algorithms are presented. Specifically, a fixed-bank MMAE and four moving-bank MMAE algorithms are discussed with some variations on the moving-bank algorithms as shown below:

1. Fixed-Bank MMAE
2. Moving-Bank MMAE Incorporating the Density Algorithm
 - Density Algorithm with Expansion and Increased Delay
3. Moving-Bank MMAE Incorporating the Density Algorithm with Sheldon Discretization
 - Density Algorithm with Sheldon Discretization, Expansion and Increased Delay
4. Moving-Bank MMAE Incorporating the Probability Algorithm
5. Moving-Bank MMAE Incorporating the Density Algorithm with Probability-Based Discretization Method

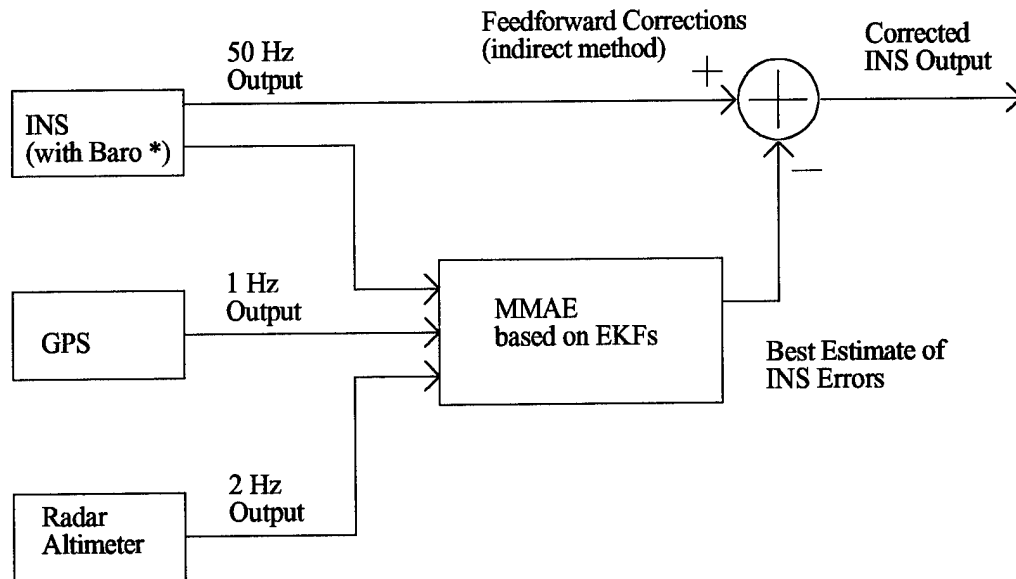
Final comparisons of the algorithms will be discussed in Chapter 5.

4.2 System Description

The following system description is a modified version of the one presented by White [77, 78] and parallels the discussion presented by Miller [53]. The 3 variations from White's discussion include deletion of a differential GPS to yield a conventional GPS, deletion of pseudolites, and reduction of the truth model from 62 to 13 states, which was a joint effort with Miller [53]. The main element of the GPS-based precision landing system (PLS) being monitored for parameter variations in its model is the GPS, specifically with respect to the amount of interference/jamming corruption (measurement noise) that enters the GPS receiver. The INS, barometric altimeter and radar altimeter also provide measurements to the Kalman filter. The following measurements are available: four satellite vehicle (SV) pseudoranges, altitude from the barometric altimeter and height above ground level from the radar altimeter.

A block diagram representing the PLS configuration is shown in Figure 37. The *true* aircraft position is generated by the trajectory profile generator PROFGEN [56] and is provided to the performance evaluation tool. The GPS satellite vehicle (SV) positions are given by actual satellite data recorded on 4 May 1991 and are combined with the true aircraft position to obtain true ranges, which are modified with appropriately modeled noise to provide *pseudoranges* measurements for use by the GPS.

Each navigation system generates measurements that are represented as perturbations from the true range, and the final *difference* measurements are then formed by subtracting the GPS measured ranges from their corresponding INS-calculated ranges. The extended Kalman filter (EKF) equations propagate estimates of the PLS error states and use the measurements to update these state estimates. Finally, these state estimates are used to correct the INS-indicated position at each sample time. The truth *and* filter models consist of 13 states (11 INS states and 2 GPS states) with details to follow.



* Barometric Altimeter incorporated into system to compensate for the fact that a bare INS has an unstable vertical channel

Figure 37. PLS Block Diagram

4.2.1 INS Models

4.2.1.1 The INS Truth and Filter Models

This section presents the truth and filter models used for the INS. The INS is a strapped-down wander azimuth system based on the Litton LN-93. The manufacturer, Litton, developed a 93-state error model [25, 62] describing the error characteristics of the LN-93. The error states $\delta \mathbf{x}$ used in the full model may be separated into 6 categories:

$$\delta \mathbf{x} = [\delta \mathbf{x}_1^T \delta \mathbf{x}_2^T \delta \mathbf{x}_3^T \delta \mathbf{x}_4^T \delta \mathbf{x}_5^T \delta \mathbf{x}_6^T]^T \quad (136)$$

where $\delta \mathbf{x}$ is a 93-dimensional column vector and:

- $\delta \mathbf{x}_1$ represents the “general” error vector containing 13 position, velocity, attitude, and vertical channel errors; the first nine states are those of the standard Pinson model [62] of INS error characteristics.
- $\delta \mathbf{x}_2$ consists of 16 gyro, accelerometer, and baro-altimeter exponentially time-correlated errors, and “trend” states. These states are modeled as first order Gauss-Markov processes.
- $\delta \mathbf{x}_3$ represents gyro bias errors. These 18 states are modeled as random constants.
- $\delta \mathbf{x}_4$ is composed of accelerometer bias error states. These 22 states are modeled in the same manner as the gyro bias states.
- $\delta \mathbf{x}_5$ depicts accelerometer and gyro initial thermal transients. The 6 thermal transient states are first-order Gauss-Markov processes with $\mathbf{w}_5 = \mathbf{0}$ so as to account for just the initial transient effects.
- $\delta \mathbf{x}_6$ models gyro compliance errors. These 18 error states are modeled as biases.

The original truth model state space differential equation is given by

$$\begin{Bmatrix} \delta \dot{\mathbf{x}}_1 \\ \delta \dot{\mathbf{x}}_2 \\ \delta \dot{\mathbf{x}}_3 \\ \delta \dot{\mathbf{x}}_4 \\ \delta \dot{\mathbf{x}}_5 \\ \delta \dot{\mathbf{x}}_6 \end{Bmatrix} = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} & \mathbf{F}_{13} & \mathbf{F}_{14} & \mathbf{F}_{15} & \mathbf{F}_{16} \\ \mathbf{0} & \mathbf{F}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{F}_{55} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \delta \mathbf{x}_1 \\ \delta \mathbf{x}_2 \\ \delta \mathbf{x}_3 \\ \delta \mathbf{x}_4 \\ \delta \mathbf{x}_5 \\ \delta \mathbf{x}_6 \end{Bmatrix} + \begin{Bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \quad (137)$$

This 93-state error model is a highly accurate LN-93 representation, but the high dimensionality of the state equation makes the model prohibitively CPU-intensive (computationally, and in terms of storage) for projects examining a large number of problem variations. The work of Negast [58] at the Air Force Institute of Technology (AFIT) addressed the reduction of the INS error-state model while preserving enough fidelity to be considered a viable truth model.

A reduced-order model is used for both the truth and filter model in this research and is defined in Equation (138):

$$\begin{Bmatrix} \delta \dot{\mathbf{x}}_1 \\ \delta \dot{\mathbf{x}}_2 \end{Bmatrix} = \begin{bmatrix} \mathbf{F}_{(red)11} & \mathbf{F}_{(red)12} \\ \mathbf{0} & \mathbf{F}_{(red)22} \end{bmatrix} \begin{Bmatrix} \delta \mathbf{x}_1 \\ \delta \mathbf{x}_2 \end{Bmatrix} + \begin{Bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{Bmatrix} \quad (138)$$

Note that the submatrix indices used in representing the 13-state model are not identical to those used in outlining the 93-state INS error model. This difference is indicated by the notation $\mathbf{F}_{(red)}$ for reduced order. The relationship between the two models is shown in Appendix D. Therefore, the INS filter model is comprised of 11 states (the first nine being the standard Pinson error model states): 3 platform misalignment errors, 3 velocity errors, 3 position errors, and 2 states for barometric altimeter stabilization.

4.2.1.2 The INS Measurement Model

The only measurement model associated directly with the INS is that for barometric altimeter aiding. The altimeter aiding is used to compensate for the instability inherent in the vertical channel of the INS. The altimeter output Alt_{Baro} is modeled as the sum of the true altitude h_t , the error in the barometric altimeter δh_B , and a random measurement noise v of variance $R_{Baro} = 3500 \text{ ft}^2$. Similarly, the INS-calculated altitude Alt_{INS} is the sum of the true altitude and the INS error in vehicle altitude above the reference ellipsoid, δh . A *difference* measurement is used to eliminate the unknown true altitude, h_t , resulting in Equation (139):

$$\begin{aligned}\delta z &= Alt_{INS} - Alt_{Baro} \\ &= [h_t + \delta h] - [h_t + \delta h_B - v] \\ &= \delta h - \delta h_B + v\end{aligned}\tag{139}$$

INS error in vehicle altitude above the reference ellipsoid, δh , and total barometric altimeter correlated error, δh_B , are states 10 and 11 in the 11-state INS model. See Appendix D for the models and numerical values of the model parameters.

4.2.2 The Radar Altimeter Model

A radar altimeter is incorporated into this application because of the intent of generating a precision landing system. A GPS-aided baro-inertial system does not have sufficient accuracy in the

vertical direction for this purpose, so the additional accurate input from a radar altimeter is used. The measurement equation of the radar altimeter is based on the difference between the INS-predicted altitude Alt_{INS} and the radar altimeter measurement Alt_{Ralt} :

$$\begin{aligned}\delta z &= Alt_{INS} - Alt_{Ralt} \\ &= [h_t + \delta h] - [h_t - v] \\ &= \delta h + v\end{aligned}\tag{140}$$

The errors in the radar altimeter are modeled as white noise with no time-correlated component. This may be a rather crude model, but should be sufficient to demonstrate performance trends. Note that no additional states are required with the addition of this radar altimeter model.

The radar altimeter measurement noise variance R_{Ralt} is a function of aircraft altitude above ground level (AGL) and will be the same in the truth and filter models. The radar altimeter noise's altitude-dependent variance [24] is given by

$$R_{Ralt} = \{[0.01]^2 * [AGL_{true}]^2\} + 0.25 \text{ ft}^2\tag{141}$$

4.2.3 GPS Models

The GPS generates user position based on “known” ranges to satellites at “known” positions. The satellites themselves transmit their position in space (in the form of ephemeris data) as accurately as it is known and the exact time (also a best estimate) at which the transmission is sent. The actual range information is calculated based on knowledge of the satellite position and the finite propagation speed of the electromagnetic radiation emitted from the satellite.

4.2.3.1 The 30-State GPS System Model

The GPS model used in this work was developed by past researchers at AFIT [16, 58, 75]. The dynamics and measurement equations for the full 30-state system model are presented in this section. Five types of error sources are modeled in the GPS state equations. The first error type,

user clock error, is common to all SV's. The remaining four error types are unique to each SV. The first two states represent user clock errors and are modeled as:

$$\begin{Bmatrix} \dot{x}_{Uclk_b} \\ \dot{x}_{Uclk_{dr}} \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} x_{Uclk_b} \\ x_{Uclk_{dr}} \end{Bmatrix} \quad (142)$$

where

$$\begin{aligned} x_{Uclk_b} &= \text{range equivalent of user clock bias} \\ x_{Uclk_{dr}} &= \text{velocity equivalent of user clock drift} \end{aligned}$$

The initial state estimates and covariances for these states were chosen to be consistent with previous AFIT research [7, 16, 58, 75] and are:

$$\begin{Bmatrix} \hat{x}_{Uclk_b}(t_0) \\ \hat{x}_{Uclk_{dr}}(t_0) \end{Bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (143)$$

and

$$\mathbf{P}_{Uclk_b, Uclk_{dr}}(t_0) = \begin{bmatrix} 9.0 \times 10^{14} \text{ ft}^2 & 0 \\ 0 & 9.0 \times 10^{10} \text{ ft}^2/\text{sec}^2 \end{bmatrix} \quad (144)$$

Because these error sources are a function of the user equipment, they are common to all the SV's. Recall that each of the remaining error types is specific to each SV, denoted by a subscript j .

The second error type is the code loop error δPR_{loop_j} . The code loop is part of the user equipment shared by all the SV's, but its error magnitude is relative to each SV. The third GPS error type is the result of atmospheric interference with the electromagnetic (EM) signals broadcast by each SV, specifically, ionospheric and tropospheric delay, δPR_{ion_j} , and δPR_{trop_j} . The code loop error, tropospheric delay, and ionospheric delay are all modeled as first-order Gauss-Markov processes with time constants shown in Equation (145). All three are driven by zero-mean white Gaussian noise with strength shown in Equation (148). The fourth error source is due to inaccuracies

of the clocks on board the individual SV's, δPR_{Sclk_j} . The final GPS error source is based on line-of-sight errors between the SV's and the receiver, δx_{s_j} , δy_{s_j} , and δz_{s_j} .

$$\begin{Bmatrix} \delta \dot{P}R_{cl_j} \\ \delta \dot{P}R_{trop_j} \\ \delta \dot{P}R_{ion_j} \\ \delta \dot{P}R_{Sclk_j} \\ \dot{\delta}x_{s_j} \\ \dot{\delta}y_{s_j} \\ \dot{\delta}z_{s_j} \end{Bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{500} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{1500} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \delta PR_{cl_j} \\ \delta PR_{trop_j} \\ \delta PR_{ion_j} \\ \delta PR_{Sclk_j} \\ \delta x_{s_j} \\ \delta y_{s_j} \\ \delta z_{s_j} \end{Bmatrix} + \begin{Bmatrix} w_{cl_j} \\ w_{trop_j} \\ w_{ion_j} \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad (145)$$

with initial covariance values given by

$$\mathbf{P}_{GPS} = \begin{bmatrix} 0.25 \text{ ft}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 \text{ ft}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 \text{ ft}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 25 \text{ ft}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25 \text{ ft}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 25 \text{ ft}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 25 \text{ ft}^2 \end{bmatrix} \quad (146)$$

(where all but the $\mathbf{P}_{GPS}(3,3)$ term have stationary characteristics. Though this value may seem strange, it was taken directly from [58]) and noise means and strengths given by

$$E[\mathbf{w}_{GPS}(t)] = \mathbf{0} \quad (147)$$

$$E[\mathbf{w}_{GPS}(t)\mathbf{w}_{GPS}^T(t+\tau)] = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.004 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.004 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ft}^2/\text{sec} \cdot \delta(\tau) \quad (148)$$

The full 30-state GPS dynamics matrix is not shown explicitly but may be easily constructed by augmenting Equation (142) and four copies (one for each SV) of Equation (145).

4.2.3.2 The GPS Truth Model and Filter Design Models

Research has shown [54,58] that the two user clock error states provide a sufficient filter model for GPS. The primary argument is that the errors modeled for the 28 other GPS states (assuming four SV's) are small when compared to the user clock errors which are common to all SV's. By increasing the dynamics driving noise and re-tuning the filter, the overall performance of the integrated navigation system can be maintained. The GPS truth and filter models used in this research are given by Equation (142) plus noise:

$$\begin{Bmatrix} \dot{x}_{Uclk_b} \\ \dot{x}_{Uclk_{dr}} \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} x_{Uclk_b} \\ x_{Uclk_{dr}} \end{Bmatrix} + \begin{Bmatrix} w_{clk_b} \\ w_{clk_{dr}} \end{Bmatrix} \quad (149)$$

4.2.3.3 The GPS Measurement Model

The pseudorange measurements available to the GPS receiver are the sum of the true range, several error sources, and a random noise:

$$PR_{GPS_j} = PR_{t_j} + \delta PR_{loop_j} + \delta PR_{trop_j} + \delta PR_{ion_j} + \delta PR_{Sclk_j} + \delta PR_{Uclk} - v_j \quad (150)$$

where

PR_{GPS_j}	=	GPS pseudorange measurement, from SV _j to user
PR_{t_j}	=	true range, from SV _j to user
δPR_{loop_j}	=	range error due to code loop error
δPR_{trop_j}	=	range error due to tropospheric delay
δPR_{ion_j}	=	range error due to ionospheric delay
δPR_{Sclk_j}	=	range error due to SV _j clock error
δPR_{Uclk}	=	range error due to user clock error
v_j	=	zero-mean white Gaussian measurement noise, $R_j = 9 \text{ ft}^2$

Because PR_t is not available to the filter, the difference between GPS pseudorange and INS-indicated pseudorange will be taken eventually to eliminate this term. First, the satellite position vector \mathbf{X}_S and the user position vector \mathbf{X}_U are defined as:

$$\mathbf{X}_U = \begin{Bmatrix} x_u \\ y_u \\ z_u \end{Bmatrix}^e, \quad \mathbf{X}_S = \begin{Bmatrix} x_s \\ y_s \\ z_s \end{Bmatrix}^e \quad (151)$$

where the superscript e denotes coordinates in the earth-centered earth-fixed (ECEF) frame. The pseudorange from the user to the satellites calculated by the INS, PR_{INS} , is the difference between the GPS/INS-calculated user position, \mathbf{X}_U , and the satellite position given by the ephemeris data, \mathbf{X}_S :

$$PR_{INS} = |\mathbf{X}_U - \mathbf{X}_S| = \left| \begin{Bmatrix} x_U \\ y_U \\ z_U \end{Bmatrix}^e - \begin{Bmatrix} x_S \\ y_S \\ z_S \end{Bmatrix}^e \right| \quad (152)$$

An equivalent form of Equation (152) is:

$$PR_{INS} = \sqrt{(x_U - x_S)^2 + (y_U - y_S)^2 + (z_U - z_S)^2} \quad (153)$$

With perturbations representing errors in \mathbf{X}_U and \mathbf{X}_S , Equation (153) can be written in terms of the true range via a truncated first-order Taylor series:

$$\begin{aligned} PR_{INS} = PR_t &+ \left. \frac{\partial PR_{INS}(\mathbf{X}_S, \mathbf{X}_U)}{\partial \mathbf{X}_S} \right|_{(\mathbf{X}_S, \mathbf{X}_U)_{nom}} \cdot \delta \mathbf{X}_S \\ &+ \left. \frac{\partial PR_{INS}(\mathbf{X}_S, \mathbf{X}_U)}{\partial \mathbf{X}_U} \right|_{(\mathbf{X}_S, \mathbf{X}_U)_{nom}} \cdot \delta \mathbf{X}_U \end{aligned} \quad (154)$$

The solution for PR_{INS} is found by evaluating the partial derivatives of Equation (153) to get:

$$\begin{aligned}
PR_{INS} = PR_t &- \left[\frac{x_S - x_U}{|PR_{INS}|} \right] \cdot \delta x_U - \left[\frac{y_S - y_U}{|PR_{INS}|} \right] \cdot \delta y_U - \left[\frac{z_S - z_U}{|PR_{INS}|} \right] \cdot \delta z_U \\
&+ \left[\frac{x_S - x_U}{|PR_{INS}|} \right] \cdot \delta x_S + \left[\frac{y_S - y_U}{|PR_{INS}|} \right] \cdot \delta y_S + \left[\frac{z_S - z_U}{|PR_{INS}|} \right] \cdot \delta z_S
\end{aligned} \tag{155}$$

Finally, the truth model GPS pseudorange *difference* measurement is given as:

$$\begin{aligned}
\delta z &= PR_{INS} - PR_{GPS} \\
&= - \left[\frac{x_S - x_U}{|PR_{INS}|} \right] \cdot \delta x_U - \left[\frac{y_S - y_U}{|PR_{INS}|} \right] \cdot \delta y_U - \left[\frac{z_S - z_U}{|PR_{INS}|} \right] \cdot \delta z_U \\
&\quad + \left[\frac{x_S - x_U}{|PR_{INS}|} \right] \cdot \delta x_S + \left[\frac{y_S - y_U}{|PR_{INS}|} \right] \cdot \delta y_S + \left[\frac{z_S - z_U}{|PR_{INS}|} \right] \cdot \delta z_S \\
&\quad - \delta PR_{loop} - \delta PR_{trop} - \delta PR_{ion} - \delta PR_{Sclk} - \delta PR_{Uclk} + v
\end{aligned} \tag{156}$$

The user position errors in Equation (156) can be derived from the first three (position error) states of the filter or truth model using an orthogonal transformation [6].

The reduced order truth and filter design measurement models for the GPS measurement do not contain terms for the errors due to code loop variations, atmospheric delays, satellite clock deviations, or errors in ephemeris-given satellite position. The filter GPS measurement model can be written as:

$$\delta z = - \left[\frac{x_S - x_U}{|PR_{INS}|} \right] \cdot \delta x_U - \left[\frac{y_S - y_U}{|PR_{INS}|} \right] \cdot \delta y_U - \left[\frac{z_S - z_U}{|PR_{INS}|} \right] \cdot \delta z_U - \delta PR_{Uclk} + v \quad (157)$$

4.3 Event Models

This section discusses the methods used to model the event changes in the computer simulations, and it parallels the discussion presented by Miller [53]. Interference/jamming is modeled as a sudden increase in the measurement noise associated with all four SV's, resulting in lower carrier-to-noise ratios, C/N_0 , in the GPS receiver. This event is induced in all SV measurements because interference/jamming is assumed to occur at the receiver, which will affect all four channels simultaneously. The interference noise variance, R_{int} , is added to the truth model measurements' R_j values (see discussion of Equation (150)) to simulate real-world interference and will be allowed to take on selected values within the interference parameter space spanned by the MMAE filter bank. Emphasis will be placed on demonstrating the capability of MMAE to detect and identify interference events of unspecified magnitude quickly. GPS *jamming* is used to refer to the total loss of useful GPS transmissions due to very large signal interference. A GPS jamming event is well-modelled (and much more easily modelled) via *very large* measurement noise. When the MMAE algorithm detects very large diagonal elements in real-world measurement noise covariance matrix \mathbf{R} , then the corresponding measurements will be *very* lightly weighted by the elemental Kalman filters; the effect is essentially the same as if those measurements were never received, hence the use of the term "interference/jamming." The system measurement noise covariance matrix is given by:

$$\mathbf{R} = \begin{bmatrix} R_{Ralt} & & & & & \\ & R_{GPS} & & & & \\ & & R_{GPS} & & & \\ & & & R_{GPS} & & \\ & & & & R_{GPS} & \\ & 0 & & & & R_{Baro} \end{bmatrix} \quad (158)$$

where R_{GPS} is the product of the nominal GPS measurement noise variance, R_0 , and either a true multiplier, a_t , or a filter-assumed multiplier, a_j .

$$R_{GPS} = \begin{cases} a_t R_0 & , \text{ for truth model} \\ a_j R_0 & , \text{ for filter models} \end{cases} \quad (159)$$

Previous research [58, 75, 77] determined reasonable choices for the nominal GPS measurement noise and the range for the noise multiplier as:

$$\begin{aligned} R_0 &= 9 \text{ ft}^2 \\ 1 &\leq a_{t,j} \leq 2000 \end{aligned} \quad (160)$$

Since the GPS measurement noise will be chosen as the scalar parameter for estimation, future notation will indicate values for the minimum and maximum allowable parameter values as:

$$\begin{aligned} a_{\min} &= 1 \\ a_{\max} &= 2000 \end{aligned}$$

Table 4 shows the six test cases used in the performance analysis. The entries in the table represent the true interference/jamming levels (multiplier values, a_t , on the four diagonal terms of \mathbf{R}_t corresponding to the GPS measurements) for each case and the time in seconds when a parameter change occurred. The simulations spanned 200 seconds of flight time (3700 – 3900 sec) out of a 2-hour flight profile generated by PROFGEN [56].

Table 4. Simulated Test Cases (Interference/Jamming Levels)

Time (sec)	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
3700	1	1	1	1	2000	1
3750	2000	1000	850	500	1000	1
3800	2000	1000	850	1000	500	1
3850	1	1	1	2000	1	1

The first three test cases represent various levels of simple interference/jamming. Case 1 induces the maximum level of interference/jamming being considered, case 2 tests the algorithms against an intermediate level of interference, while case 3 was added to provide a fair comparison of the algorithms. This will be discussed in detail later, but the basic idea is that case 2 induces an interference level and thus a true parameter value that almost perfectly matches one of the filter-

assumed parameter values in the fixed-bank MMAE. It is also important to select a true parameter value that does not match any of the fixed-bank filter-assumed parameter values in order to compare the performance of the fixed-bank MMAE against the moving-bank algorithms properly. Cases 4 and 5 allow analysis of the algorithms in the presence of relatively small parameter changes and identify the differences in performance when the parameter value is increasing versus decreasing. Finally, case 6 provides a baseline indicating the level of state estimation performance achieved in the environment without any interference/jamming corruption.

The simulated flight profile being implemented is illustrated in Figure 38. Notice the continual drop in altitude toward ground level, indicating that the landing portion of the flight profile was used. A fixed-bank MMAE and four moving-bank MMAE algorithms discussed further in Sections 4.6 – 4.10 were analyzed for each of the six test cases. This will provide results for a conventional MMAE (see Figure 6 on page 8) used to provide state and parameter estimates simultaneously. However, all the algorithms have been tuned and otherwise adjusted (such as the selection of threshold values) to enhance *parameter* estimation, leaving precise state estimations as a secondary objective. This includes applying the modified Sheldon discretization optimized for parameter estimation in lieu of the discretization optimized for state estimation, in cases where a Sheldon discretization is performed. The motivation for focusing on parameter estimation is the intent to utilize the M³AE architecture (see Figure 7 on page 9) for the final state estimation. Fortunately, all the data from both the conventional MMAE and the M³AE is available; so analysis will be conducted on both. Many of the case studies produce similar trends, so the discussion of some cases will be brief. However, each case presented did identify at least one strength or weakness of a given algorithm and will be presented in the section relative to that algorithm. Although some comparisons will be drawn between algorithms, final conclusions will be stated in Chapter 5.

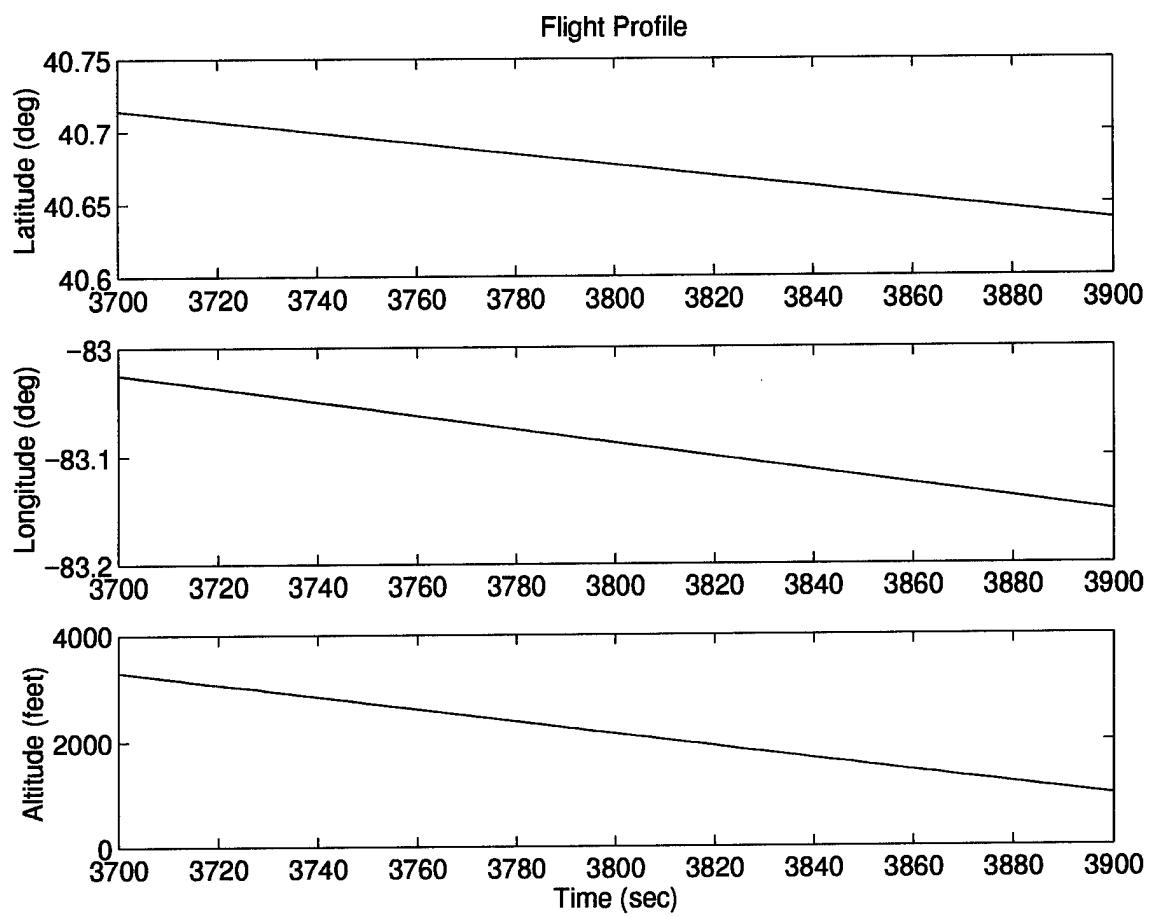


Figure 38. Simulated Flight Profile

4.4 Data Presentation

The simulation data is presented through four types of plots and a tabular listing of two performance measures. First, the state plots of aircraft latitude, longitude and altitude are plotted for the blended estimates provided by the MMAE and for the single filter generating the final state estimates via the M³AE architecture. Each plot contains five traces. The innermost trace (—) on each data plot is the *mean error* time history for the applicable state. Mean Error is defined as the difference between the filter's estimate of the state and the true state, averaged over the number of Monte Carlo runs performed. The equation describing this relationship is defined by [44]:

$$\hat{M}_e(t_i) = \frac{1}{N} \sum_{\omega=1}^N e_{\omega}(t_i) = \frac{1}{N} \sum_{\omega=1}^N \{\hat{x}_{\omega}(t_i) - x_{true_{\omega}}(t_i)\} \quad (161)$$

where $\hat{x}_{\omega}(t_i)$ is the filter-computed estimate of a given state and $x_{true_{\omega}}(t_i)$ is the *truth* model value of the same state, at time t_i , for run ω , and N is the number of time histories in the simulation (10 in this dissertation).

In addition to the center trace, two more *pairs* of traces are plotted and identified as the state estimate error *Mean* \pm *Sigma* ($\mu_x \pm \sigma_x$). The first pair (represented by $\cdot - \cdot -$) is symmetrically displaced about the mean and as a result follows the “undulations” of $\hat{M}_e(t_i)$. The locus of these traces is calculated from $\hat{M}_e(t_i) \pm \sqrt{P_e(t_i)}$, where $P_e(t_i)$ is the *true* error variance at time t_i . The true standard deviation is calculated from [44]:

$$\sigma_{true}(t_i) = \sqrt{P_e(t_i)} = \sqrt{\frac{1}{N-1} \sum_{\omega=1}^N e_{\omega}^2(t_i) - \frac{N}{N-1} \hat{M}_e^2(t_i)} \quad (162)$$

where N is the number of runs in the Monte Carlo simulation (10 in this case), and $\hat{M}_e^2(t_i)$ is the square of the mean of a given state at each time of interest. The last pair of traces (—) represent the *filter-computed* $\pm\sigma_{filter}$ values for the same states and are symmetrically displaced about zero because the filter “believes” that it is producing zero-mean errors [44]. These traces represent the filter’s estimate of its own error. Thus, on a single plot, the $[\hat{M}_e(t_i) \pm \sqrt{P_e(t_i)}]$ traces provide an indication of *true performance*, while the $[\pm\sigma_{filter}]$ traces additionally show how well tuned the filter is, i.e., how well it predicts its own errors.

The second type of plot presents the filter probabilities over the time history for a *single representative* run. Alternatively, the mean and standard deviation of the probabilities could be presented, but single run plots provide insights into trends that could be smoothed over or averaged out by the statistics of the multi-run data. Similarly, single run data may *not* fully represent the performance being analyzed if there are large variations from run to run. Therefore, all of the Monte Carlo runs were observed in the selection of this single run to ensure it adequately represents the overall performance. Also, a single example of the mean and standard deviation of the probabilities is presented to illustrate that significant variation of the probability data does not exist from run to run. The ordinate is labeled EF1 – EF5 to identify the elemental filter number.

The third type of plot shows the time history for the parameter being estimated in this problem for a *single representative* run. The first trace (\cdots) represents the *true* parameter value as dictated by the case study being simulated. The second trace (—) shows the parameter estimate provided by the MMAE. The third trace (- - -) indicates the minimum and maximum filter-assumed parameter values within the MMAE bank at each sample time. This *pair* of traces illustrates the breadth of the bank, how well the bank encompasses the true parameter value and where the parameter estimate lies within the MMAE bank, i.e., close to an endpoint or somewhere in the middle. Alternatively, the mean and standard deviation of the parameter data based on all the Monte Carlo runs could be

presented, but the plots would be cluttered and difficult to interpret. Therefore, single representative runs are presented for these plots with a combination of parameter data, and statistics for the parameter estimates are shown in the fourth type of plot.

The fourth type of plot shows statistics of the parameter estimate error. The innermost trace (—) on each data plot is the *mean error* time history for the parameter estimate. Mean Parameter Estimate Error, $\hat{M}_{e_a}(t_i)$, is defined as the difference between the MMAE blended estimate of the parameter and the true parameter, averaged over the number of Monte Carlo runs performed. The equation describing this relationship is defined by [44] and is identical to Equation (161) replacing $e_\omega(t_i) = \{\hat{x}_\omega(t_i) - x_{true_\omega}(t_i)\}$ with $e_{\omega_a}(t_i) = \{\hat{a}_{MMAE_\omega}(t_i) - a_{true_\omega}(t_i)\}$. Similarly, the trace pair ($\cdot - \cdot$) represents the parameter estimate error $\hat{M}_{e_a}(t_i) \pm \sqrt{P_{e_a}(t_i)}$ and utilizes Equation (162), replacing $e_\omega(t_i)$ with $e_{\omega_a}(t_i)$ and $\hat{M}_e(t_i)$ with $\hat{M}_{e_a}(t_i)$.

The performance measure used to assist in analyzing the state plots is the temporally averaged RMS value of the state estimation error and is given by:

$$\hat{e}_{RMS}^x = \frac{1}{N_{samp}} \sum_{i=0}^{N_{samp}} \sqrt{\frac{1}{N} \sum_{\omega=1}^N e_\omega^2(t_i)} \quad (163)$$

where N_{samp} is the number of time samples of the state estimation error. Similarly, the performance measure used for parameter estimation analysis, \hat{e}_{RMS}^a , is the temporally averaged RMS value of the parameter estimation error given by Equation (163) but with $e_\omega(t_i)$ replaced by $e_{a_\omega}(t_i) = [\hat{a}_{MMAE_\omega}(t_i) - a_{true_\omega}(t_i)]$.

4.5 Simulation Software

Multimode Simulation for Optimal Filter Evaluation (MSOFE) is a general-purpose, multimode simulation program for designing integrated systems that employ optimal (Kalman) filtering

techniques and for evaluating their performance [57]. The general-purpose construction of MSOFE allows its application to a wide variety of user-specific problems with a minimal amount of new software development. The United States Air Force uses MSOFE for the validation of systems that use optimal filtering techniques. MSOFE provides Monte Carlo and covariance simulation modes.

Multiple Model Simulation for Optimal Filter Evaluation (MMSOFE) was developed at AFIT by Nielsen [59,60] to support the analysis of systems using a multiple model adaptive filter structure. MMSOFE is written as an extension to MSOFE and is based on the same core code. The MMSOFE program propagates multiple filters forward in parallel while performing the hypothesis probability and blending calculations required for MMAE and other multiple model algorithms. The Monte Carlo simulation mode of MMSOFE with 10 runs ($N = 10$) is used in all phases of the work.

4.6 Fixed Bank MMAE

4.6.1 Implementation Issues

A fixed-bank MMAE was simulated for comparison to the various moving-bank MMAE methods investigated in this chapter. The basic concept of the fixed-bank MMAE was presented in Section 2.2. A bank of 5 elemental filters was implemented, and the modified Sheldon discretization optimized for parameter estimation with a finite horizon of 10 samples = 10 seconds was used to determine the parameter values for the filters. The filter-assumed measurement noise covariance matrix is given by:

$$\mathbf{R}_j = \begin{bmatrix} R_{Ralt} & & & & \\ & R_{GPS} & & & \\ & & R_{GPS} & & \\ & & & R_{GPS} & \\ & 0 & & & R_{GPS} \\ & & & & & R_{Baro} \end{bmatrix}$$

where R_{Ralt} is given by Equation (141), $R_{Baro} = 3500 ft^2$ as stated in Section 4.2.1.2, and R_{GPS} is the product of the nominal GPS measurement noise covariance, $R_0 = 9 ft^2$ (see Section 4.3), and

the filter-assumed multiplier, a_j .

$$R_{GPS} = a_j R_0$$

The resulting multipliers of the nominal GPS measurement noise covariance are shown in Table 5. Notice that filter 1 has a noise multiplier of $a_j = 1$. It makes sense to assign a filter the parameter value associated with the nominal case of no interference/jamming; therefore, the noise multiplier for filter 1 was preassigned when performing the Sheldon discretization. Figure 39 shows the estimation error autocorrelation curve generated by the Sheldon algorithm for this problem. The parameter values determined by the Sheldon algorithm correspond to the intersection points of the autocorrelation curve with the abscissa. Lower bounding of the probabilities was implemented as discussed in Section 2.2.2. Specifically, a lower bound of $p_{\min} = 0.001$ was used and applied in every implementation presented in this research.

Table 5. Parameter Values Chosen via Sheldon

Filter j	1	2	3	4	5
Noise Multiplier, a_j	1	188	615	1097	1660

4.6.2 Performance

Case 1 (recall Table 4 on page 149): The true and estimated parameter plot shown in Figure 40 identifies a potential liability of the fixed-bank algorithm when the true parameter value approaches the maximum admissible parameter value. Notice that, although the true parameter multiplier, a_t , equals $a_{\max} = 2000$ from time $3750 \text{ sec} \leq t < 3850 \text{ sec}$, the parameter estimate, \hat{a}_{MMAE} , is consistently lower and bounded above by the largest filter-assumed parameter value of $a_5 = 1660$. This upper bound is shown by the trace (---), and the lower bound trace (---) is hidden along the bottom edge of the plot. One solution to this problem would be to set $a_5 = a_{\max}$, but recall that a_5 was determined via a Sheldon discretization over the parameter range a_{\min} to a_{\max} . Artificially setting $a_5 = a_{\max}$ would negate the benefits of using Sheldon's cost minimization technique for

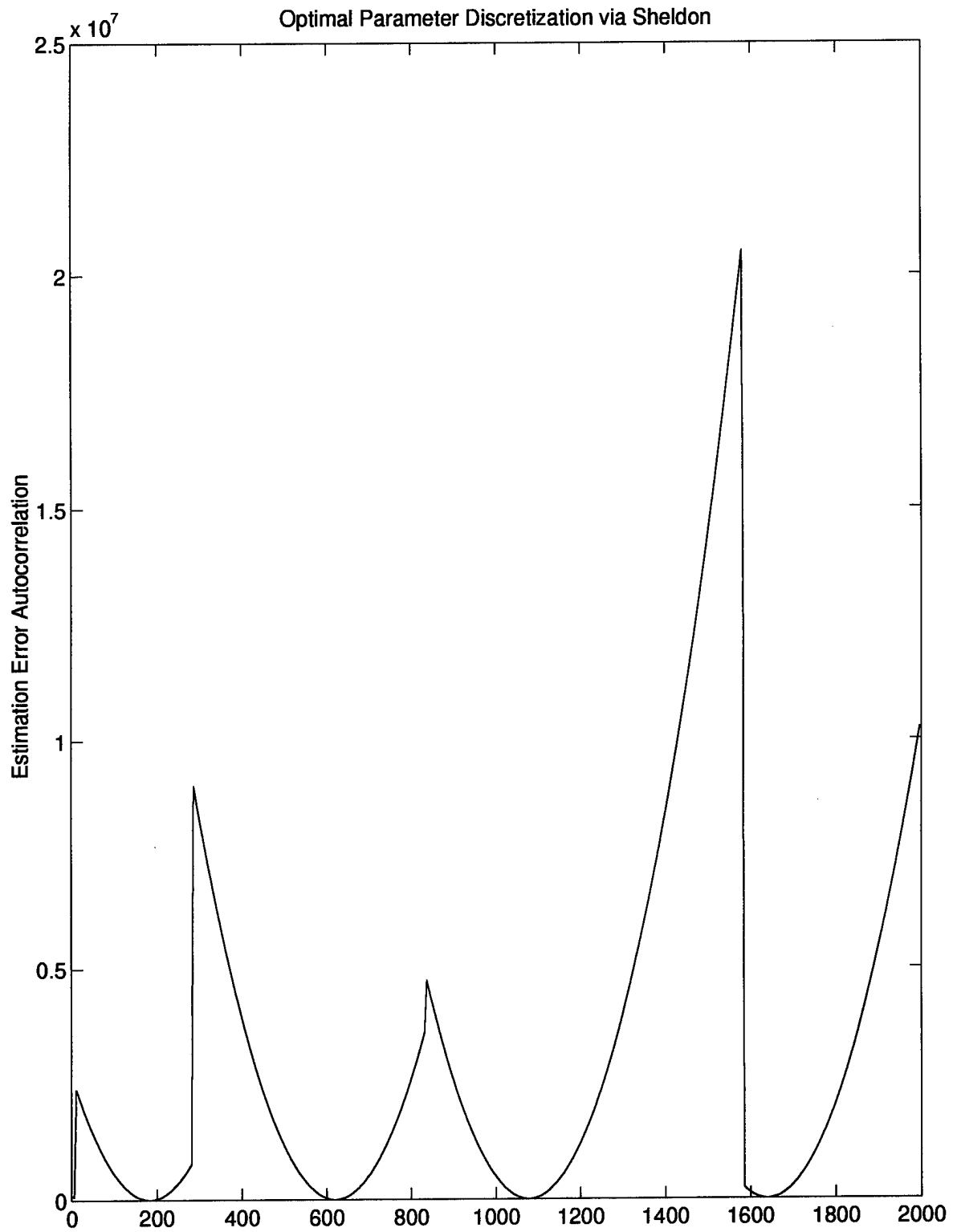


Figure 39. Autocorrelation Curve for Constrained-Range Parameter Discretization

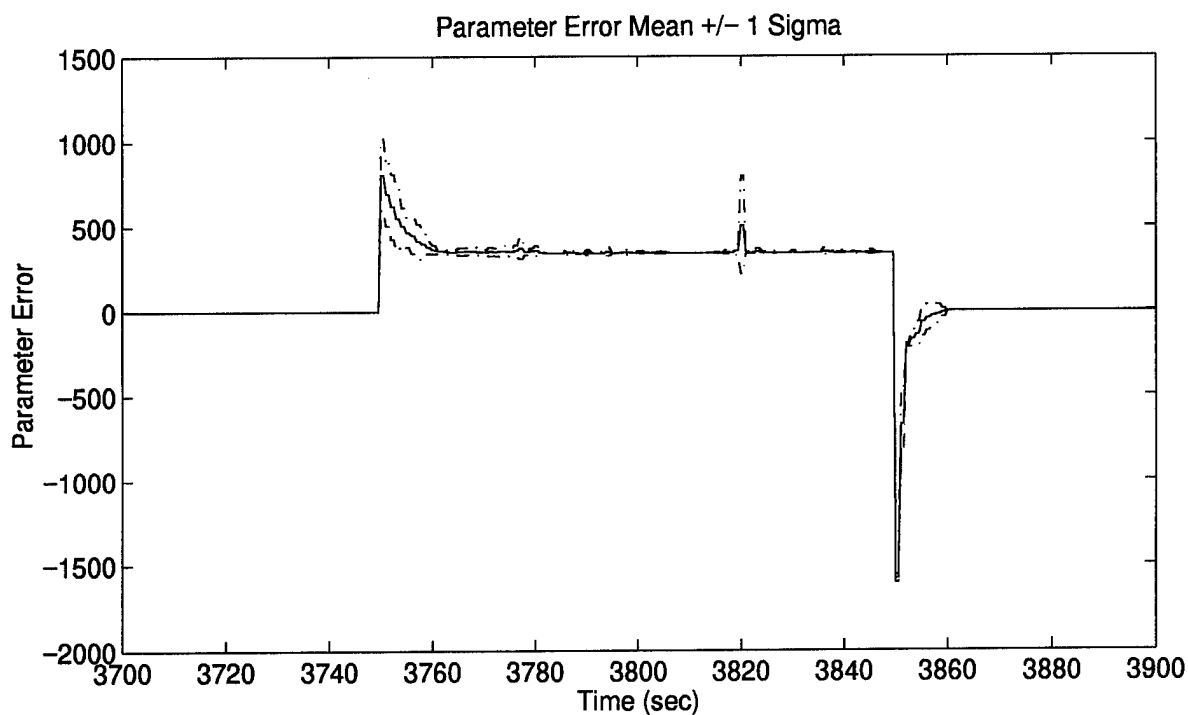
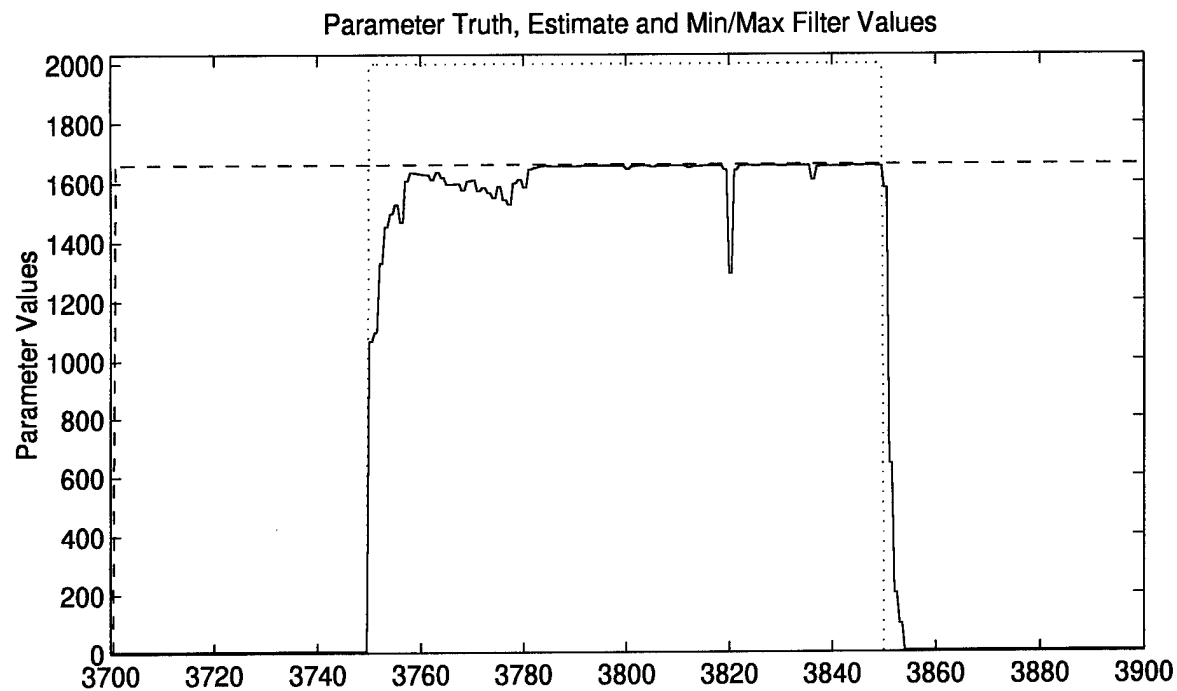


Figure 40. Parameter Estimation Performance – Case 1: Fixed-Bank

minimal parameter estimation error, as would artificially increasing a_{\max} prior to implementing Sheldon's algorithm such that $a_5 \simeq 2000$. The result of underestimating the true parameter value is illustrated by the relatively large value for the parameter estimation measure, $\hat{e}_{RMS}^a = 203$, shown in Table 16 of Appendix E.

The *blended* state estimate errors, $\hat{\mathbf{x}}_{\text{MMAE}}$, provided by the conventional MMAE are shown in Figure 41. All three states show the quick response of the MMAE to the abrupt change in the true parameter at $t = 3750$ sec and 3850 sec. The altitude error state plot indicates adequate tuning in this channel. In particular, the filter computed standard deviation, $\pm\sigma_{\text{filter}}$, encompasses the majority of the error state $\mu_x \pm \sigma_x$ values (with a notable exception at $t = 3850$ sec, at the time of the second abrupt change in the true parameter value). An important characteristic of the altitude error state plots is the obvious "tapering down" of both the $\mu_x \pm \sigma_x$ and $\pm\sigma_{\text{filter}}$ values over time. Recall the gradual descent in altitude shown in Figure 38, and the dependence of the radar altimeter's noise variance on this altitude, as seen in Equation (141). This combination results in radar altimeter measurements which are continually improving over time in terms of their accuracy (i.e., the radar altimeter becomes more effective as it approaches ground level, and its associated measurement noise variance is decreasing). Also, note that the size of the "sawtooth" created by the $\pm\sigma_{\text{filter}}$ values increases over time. This increase implies that the measurement update is having a larger (positive) impact on the error state estimates and makes sense since a decreased measurement noise variance infers an increased reliance on the measurement data. These accurate radar altimeter measurements directly improve the altitude state estimation performance, as seen in the results throughout this chapter. The latitude and longitude error states are also well tuned in the time frames of $t < 3750$ sec and $t > 3850$ sec (without interference/jamming) with the exception of some transient behavior around $t = 3850$ sec. These transients result from the time needed by the MMAE to converge on the true parameter value. The filter probabilities plotted in Figure 42 indicate that 5

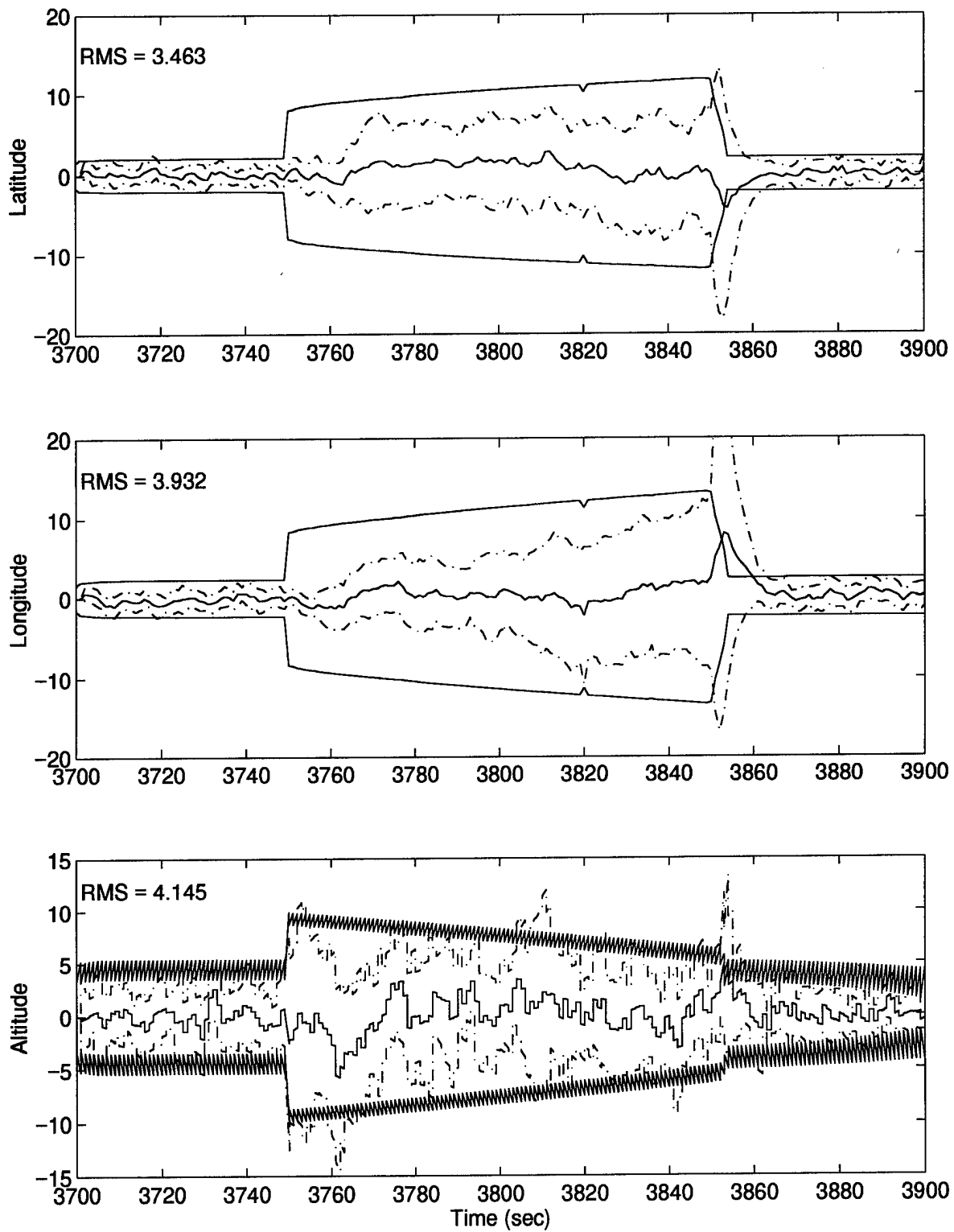


Figure 41. MMAE State Estimation Errors (feet) – Case 1: Fixed-Bank

sample times (3850 sec to 3854 sec) are needed before significant probability weight transfers from filter 5 to filter 1, i.e., $p_5 = 0.9948 \rightarrow 0.001$ while $p_1 = 0.001 \rightarrow 0.9954$. During this transfer time, the error state estimates degrade, resulting in an increase in the relative mean and variance values. Additionally, once the MMAE has converged on the best filter-assumed parameter match to truth, several more sample periods are required for the state estimate errors to converge back to a zero-mean condition.

Although the probability plot is only shown for 1 of the 10 Monte Carlo runs, this trend is consistent across all the runs, as illustrated by the mean and mean \pm one standard deviation of the probabilities shown in Figure 43. Comparison of Figures 42 and 43 indicates that the single run data are representative of the mean probability values (trace —), and the relatively small standard deviations (trace $\cdot - \cdot -$) in Figure 43 indicate minor variations in the data from run to run. The latitude and longitude error states are somewhat conservatively tuned (i.e., the *filter-computed* $\pm \sigma_{filter}$ values overestimate the *true* $\mu_x \pm \sigma_x$ values) when the interference/jamming is present ($3750 \text{ sec} \leq t < 3850 \text{ sec}$) for both the MMAE and M³AE error state estimates (see Figures 41 and 44). Figure 40 shows that $\hat{\mathbf{a}}_{\text{MMAE}}$ is biased low with respect to \mathbf{a}_t (i.e., the measurement noise covariance is estimated lower than truth), yet the M³AE error state estimates indicate conservative tuning of the single Kalman filter being sent these parameter estimates. This would make sense if the parameter estimate were biased high such that the single Kalman filter assumes too high a value for the covariance values associated with the GPS measurement noise. This conservatism will be evident in many case studies for many of the algorithms, so further discussion will be presented here with emphasis on the M³AE error state estimates given below. The relatively benign flight environment is seen from the flight profile in Figure 38. Notice that the latitude and longitude are gradually changing and monotonic, resulting in accurate representation of the system dynamics by the filter models. Given a more turbulent flight profile *and* filter dynamics models *not* adequately modified

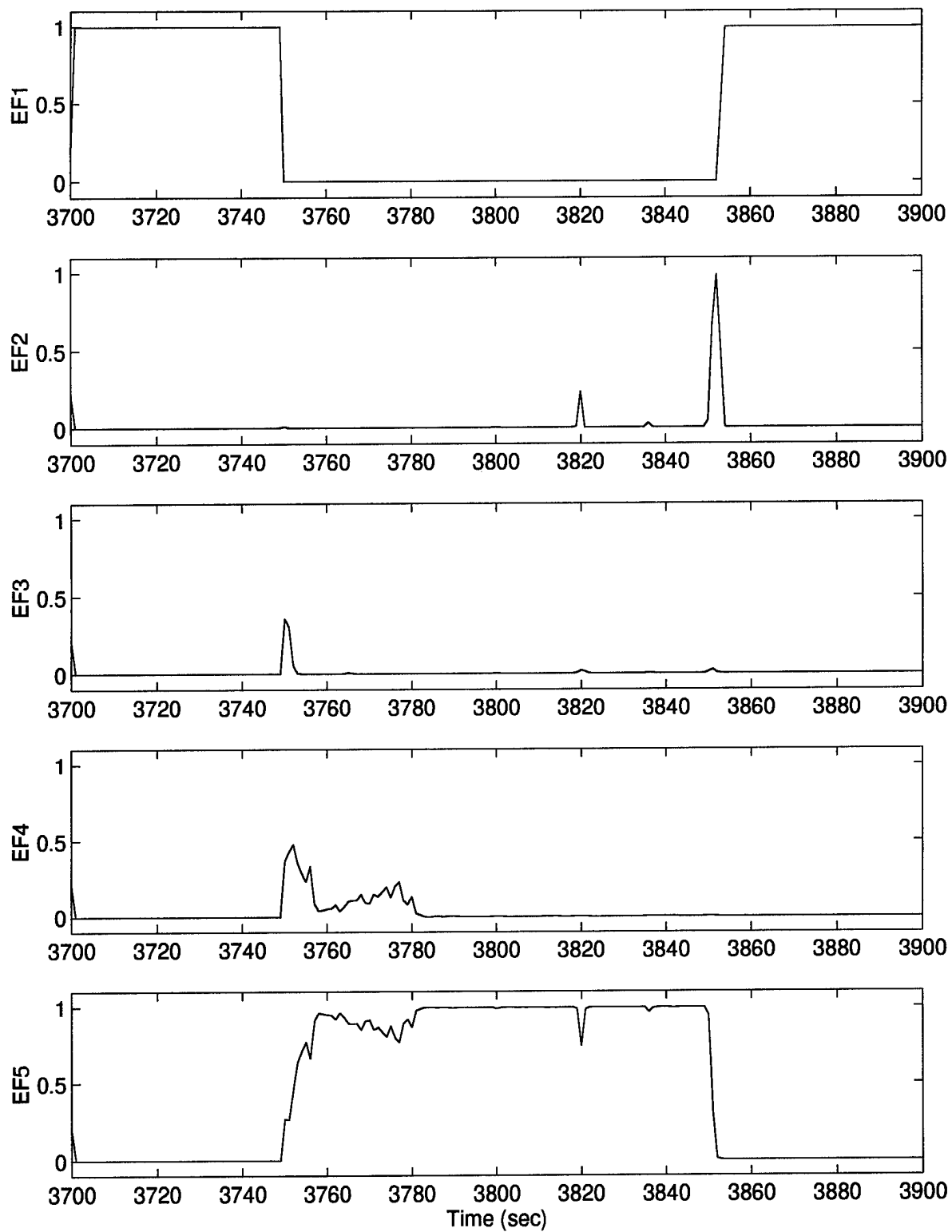


Figure 42. Elemental Filter Probabilities – Case1: Fixed-Bank

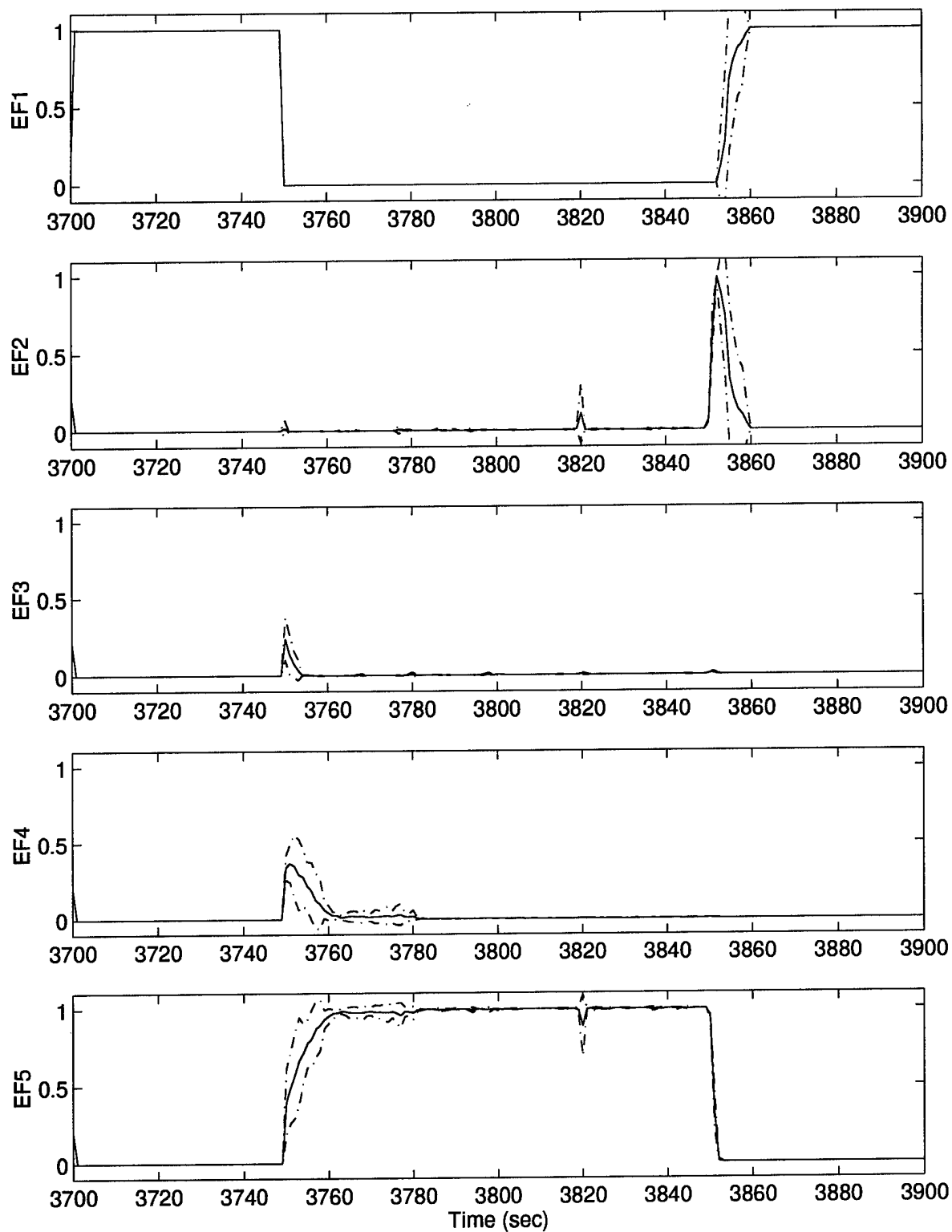


Figure 43. Elemental Filter Probabilities Mean ± 1 Sigma – Case 1: Fixed-Bank

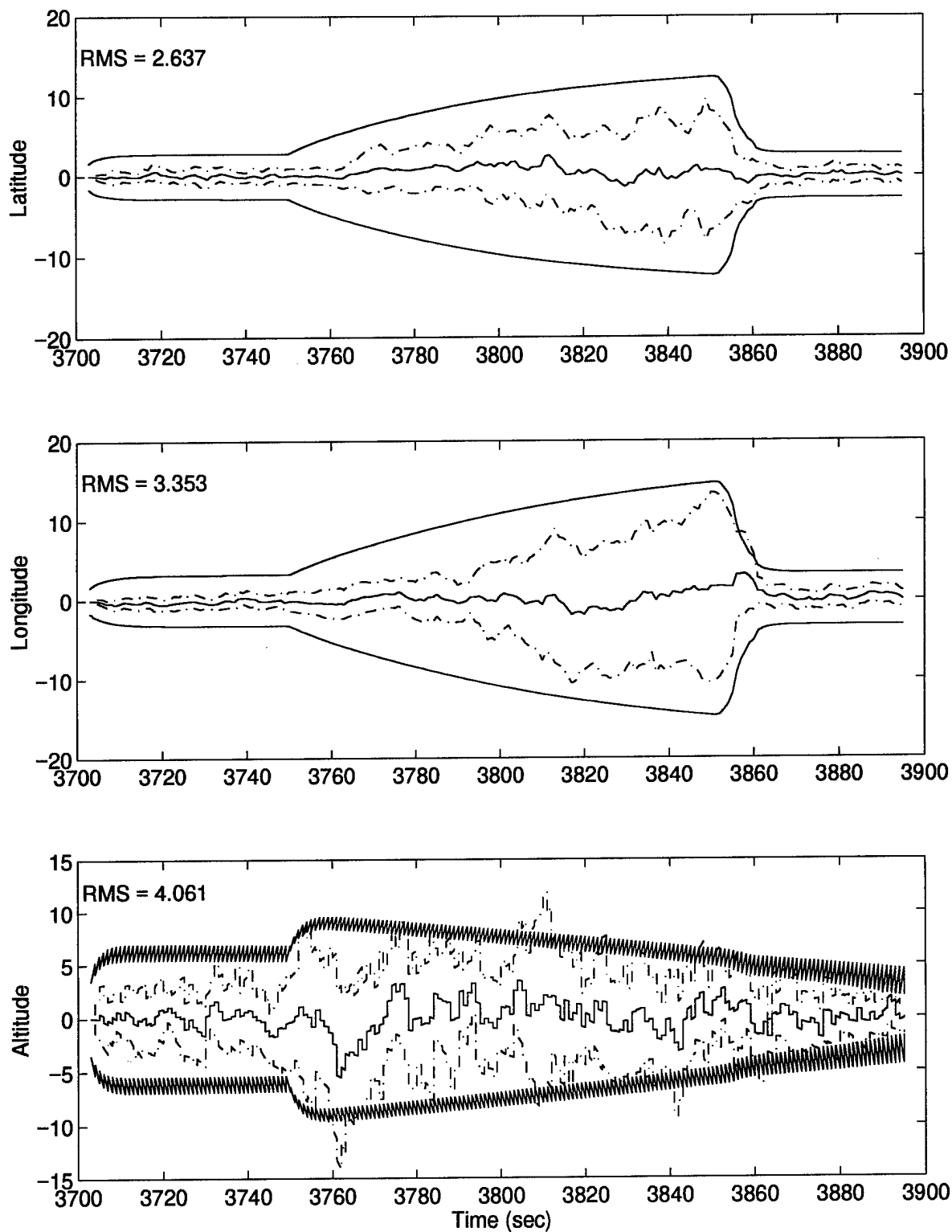


Figure 44. M³AE State Estimation Errors (feet) – Case 1: Fixed-Bank

to account for this high level of turbulence, the dynamics models in the filters *could* become less accurate. However, the scenario in this simulation involves a less turbulent profile and a large increase in the measurement noise covariance values, resulting in more dependence on the *accurately modeled* dynamics models and less dependence on the incoming measurements.

This can be seen mathematically from the Kalman filter propagate and update equations repeated here.

$$\hat{\mathbf{x}}(t_i^-) = \Phi(t_i, t_{i-1})\hat{\mathbf{x}}(t_{i-1}^+) + \mathbf{B}_d(t_{i-1})\mathbf{u}(t_{i-1}) \quad (164)$$

$$\mathbf{K}(t_i) = \mathbf{P}(t_i^-)\mathbf{H}^T(t_i) [\mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i)]^{-1} \quad (165)$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)[\mathbf{z}_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)] \quad (166)$$

Notice that, as \mathbf{R} gets larger, as in the interference/jamming case, the gains \mathbf{K} decrease, resulting in less weight on the measurements, \mathbf{z} , and more reliance on the propagated state estimates, $\hat{\mathbf{x}}(t_i^-)$, in determining the updated state estimates, $\hat{\mathbf{x}}(t_i^+)$. Furthermore, since the filter and truth models are the same with the exception of the noise parameter being estimated, the need for accurate measurements is lessened versus the case in which the filter models are a reduced-order version of the truth model. Therefore, although error states 1 and 2 do not appear particularly well tuned, this does not result in significant degradation of their estimates and large variations in the estimation errors.

The *final* error state estimates provided by the M³AE architecture are shown in Figure 44. The error state estimation measure, \hat{e}_{RMS}^x , shown in Tables 17 – 22 of Appendix E, indicates a considerable improvement in the state estimates resulting from the M³AE approach. This is primarily due to the reduction of the transient behavior around $t = 3850$ sec. The parameter estimate converges toward truth within a few sample periods, allowing the single filter, which uses this estimate, to adapt its measurement noise covariance tuning quickly, resulting in smaller standard deviations for the error state estimates. Observation of the M³AE error state estimates reveals conservative tuning present from $3700 \text{ sec} \leq t < 3750 \text{ sec}$. This is a result of the parameter being estimated much higher

than truth ($\hat{\mathbf{a}}_{\text{MMAE}} > \mathbf{a}_t$), as seen in Figure 45. The probability lower bounding requires that all the elemental filters, even those that are clearly far from truth, contribute to the blended parameter estimate given by:

$$\hat{\mathbf{a}}_{\text{MMAE}}(t_i) = \sum_{j=1}^J \mathbf{a}_j p_j(t_i)$$

Since some of the filters assume large parameter values, \mathbf{a}_j , the blended estimate is biased high in the unjammed environment and causes the conservative tuning of the M³AE estimates. A simple solution to this conservative tuning will be presented in Section 4.11.

Now that the conservative nature of the M³AE state estimates is understood when $\hat{\mathbf{a}}_{\text{MMAE}} > \mathbf{a}_t$, a second issue stems from the M³AE error state estimates being *more* conservatively tuned than their associated MMAE error state estimates. Recall that the M³AE error state estimates are functions of $\hat{\mathbf{a}}_{\text{MMAE}}$ which is passed from the MMAE to the single Kalman filter. More precisely, the M³AE error state estimates are generated by a Kalman filter which assumes a parameter value equal to $\hat{\mathbf{a}}_{\text{MMAE}}$. Furthermore, $\hat{\mathbf{a}}_{\text{MMAE}}$ is a function of the elemental filter probabilities, p_j , and the filter-assumed parameter values, \mathbf{a}_j . Similarly, the MMAE error state estimates, $\hat{\mathbf{x}}_{\text{MMAE}}$, are functions of these *same* filter probabilities, p_j , and the error state estimates generated by each of the elemental Kalman filters, $\hat{\mathbf{x}}_j$. The natural question that arises is, “Given that both architectures utilize the same filter probabilities, p_j , why are the M³AE error state estimates *more* conservatively tuned than their associated MMAE error state estimates?” The research by Miller [53] capitalized on the fact that these two sets of state estimates would *not* be the same and used this as a motivation for implementing the M³AE architecture. Particularly, for the case of coarse discretization, a *single* Kalman filter based on a good parameter estimate, $\hat{\mathbf{a}}_{\text{MMAE}}$, is superior to any linear combination of state

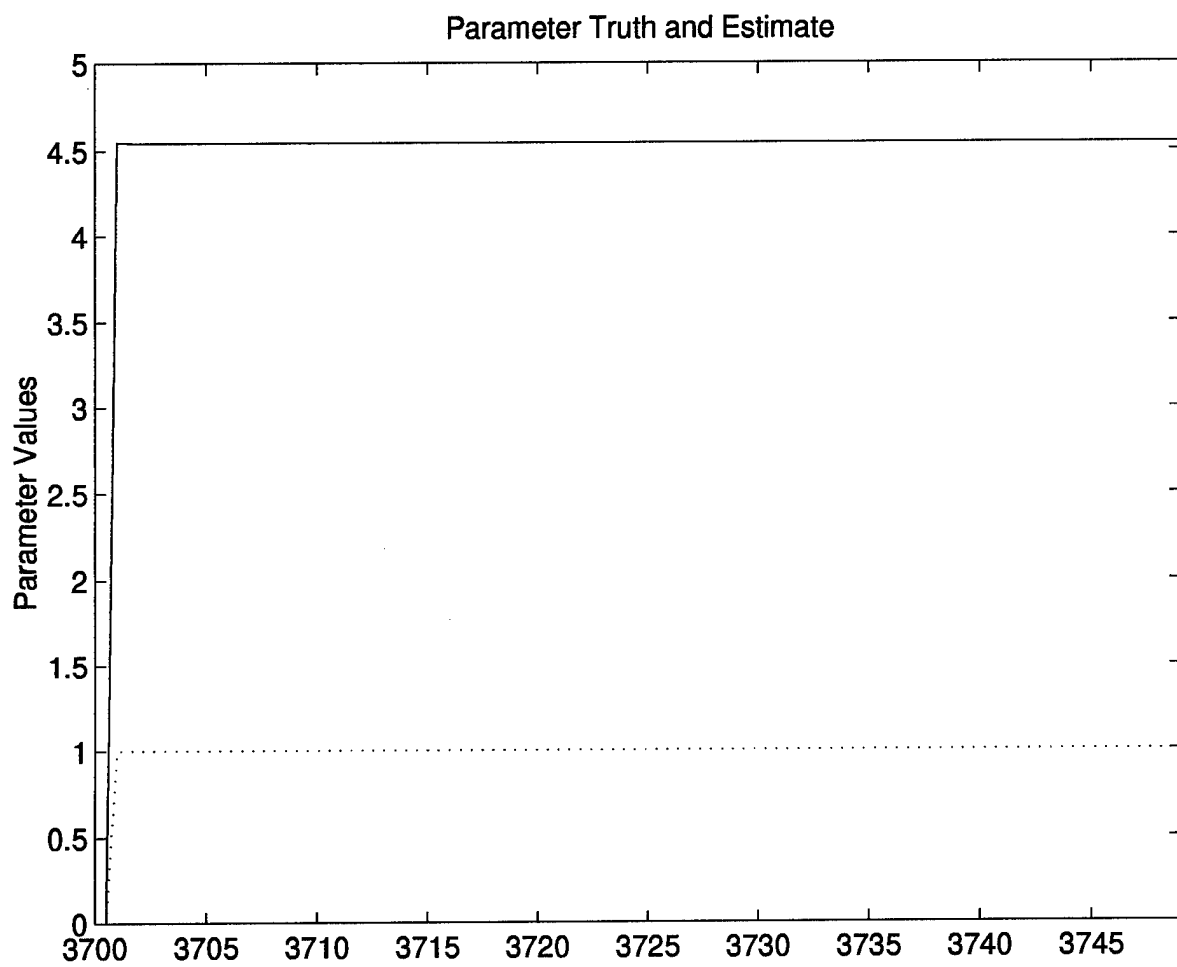


Figure 45. Parameter Estimate ($3700 \text{ sec} \leq t < 3750 \text{ sec}$) – Case 1: Fixed-Bank

estimates from a bank of Kalman filters, all of which are based on poor a_j values. This also relates directly to the observation that M^3AE greatly outperforms $MMAE$ *only* when there is significant blending of more than one elemental filter to produce \hat{a}_{MMAE} (thereby very different from any of the a_j 's currently in the bank). However, the question raised above has yet to be addressed and is a natural extension to Miller's research [53].

Further clarification is needed for the performance of the M^3AE error state estimates during the time frame $3750 \text{ sec} \leq t < 3850 \text{ sec}$, in which the parameter estimate is less than truth (i.e., $\hat{a}_{MMAE} \simeq 1655$ while $a_t = 2000$), and a bias high of the parameter estimate cannot be identified as the cause for the conservative tuning. Typically, underestimating the measurement noise covariance would result in a *less* conservatively tuned filter, and the filter-computed $\pm \sigma_{filter}$ values would underestimate the true $\mu_x \pm \sigma_x$ values. The previous paragraphs explained that the conservatism was due to a less turbulent profile and a large increase in the measurement noise covariance values. This concept still holds here despite the underestimation of the true parameter value. The reason is that, with $a_t = 2000$, the GPS measurements have been effectively eliminated, and the previous discussion identified that the error state estimates (for states 1 and 2) become a function of the dynamics model. Again, significant degradation of the error state estimates does *not* occur in this benign flight profile with the exception of a gradual increase in the true error variance resulting from INS drift (i.e., the GPS is not providing any feedforward corrections to the INS). Similarly, with $\hat{a}_{MMAE} \simeq 1655$, which is a large underestimation of $a_t = 2000$, the GPS measurements have still been effectively eliminated and the filter tuning appears conservative. Additional simulations (not presented here) were conducted with very small interference levels such as $a_t = 10$ to verify that this conservatism could be removed in cases in which the resulting \hat{a}_{MMAE} was less than a_t but not so large as to eliminate the effects of the GPS measurements.

Case 2: Additional insights into the performance of the fixed-bank algorithm are gained by observation of the parameter estimate and parameter error plots for this case; recall from Table 4 (page 149) that the step change is from 1 to 1000 on the parameter, rather than 2000 as in the previous case. The plots of the error state estimates look essentially the same as in case 1, and the only discernible difference is a slight reduction in the performance measure \hat{e}_{RMS}^x for all three states. Figures 46 and 47 illustrate the interaction between elemental filters 3 and 4 in the tracking of a_t .

First, the parameter estimate seems somewhat erratic from $t = 3750$ sec to 3795 sec, then settles on a value that is biased high with respect to a_t . The parameter error mean ± 1 standard deviation plot reinforces this observation with a mean value for the error that is rather dynamic immediately after the interference/jamming is induced but essentially constant after $t = 3795$ sec. Similarly, the standard deviation is initially large, then shrinks considerably once the mean value settles down to a constant value. The probability plots help explain the results above by showing how filters 3 and 4 compete for the probability weight from $t = 3750$ sec to 3795 sec, followed by convergence to the more conservatively tuned filter (EF4). Recall that filters 3 and 4 have assumed parameter values of 615 and 1097, respectively, and the true interference/jamming level is $a_t = 1000$ for this case study. Therefore, the fixed-bank performance which converges to filter 4 makes sense. Additionally, having an elemental filter assume a parameter value that is very close to the true parameter value gives the appearance of superior parameter estimation, as seen by comparing the values of \hat{e}_{RMS}^a in Table 16 (Appendix E) for each algorithm. This no longer holds when the true parameter value is not particularly close to any one filter, as seen in case 3 below.

Case 3: In contrast to case 2, the true parameter value ($a_t = 850$) is halfway between the filter-assumed parameter values for filters 3 and 4, resulting in a degraded parameter estimate relative to that of case 2. See the increased standard deviation of the parameter error in Figure 48 and the increased measure values \hat{e}_{RMS}^a and \hat{e}_{RMS}^x in Tables 16 – 22 (Appendix E). The primary problem

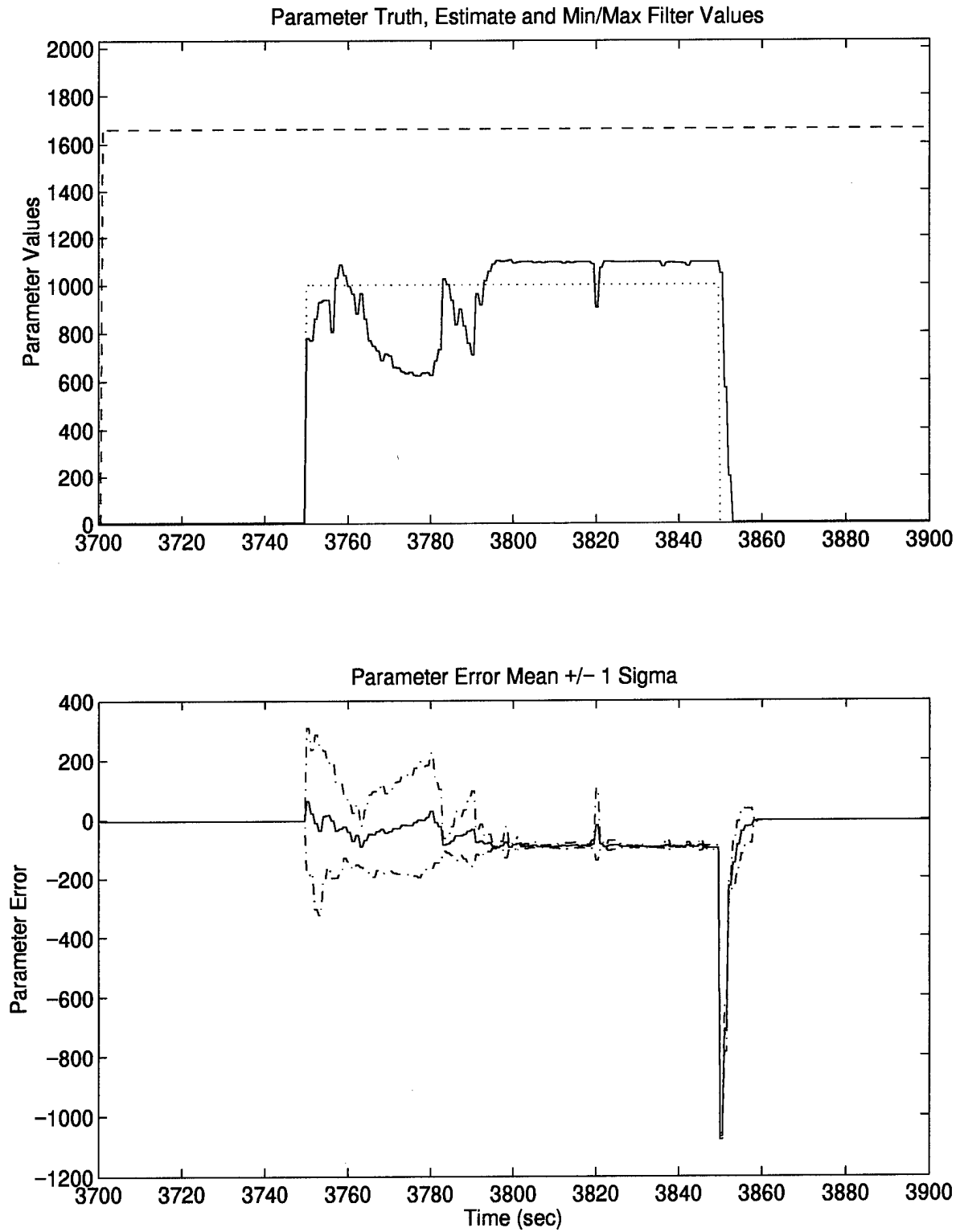


Figure 46. Parameter Estimation Performance – Case 2: Fixed-Bank

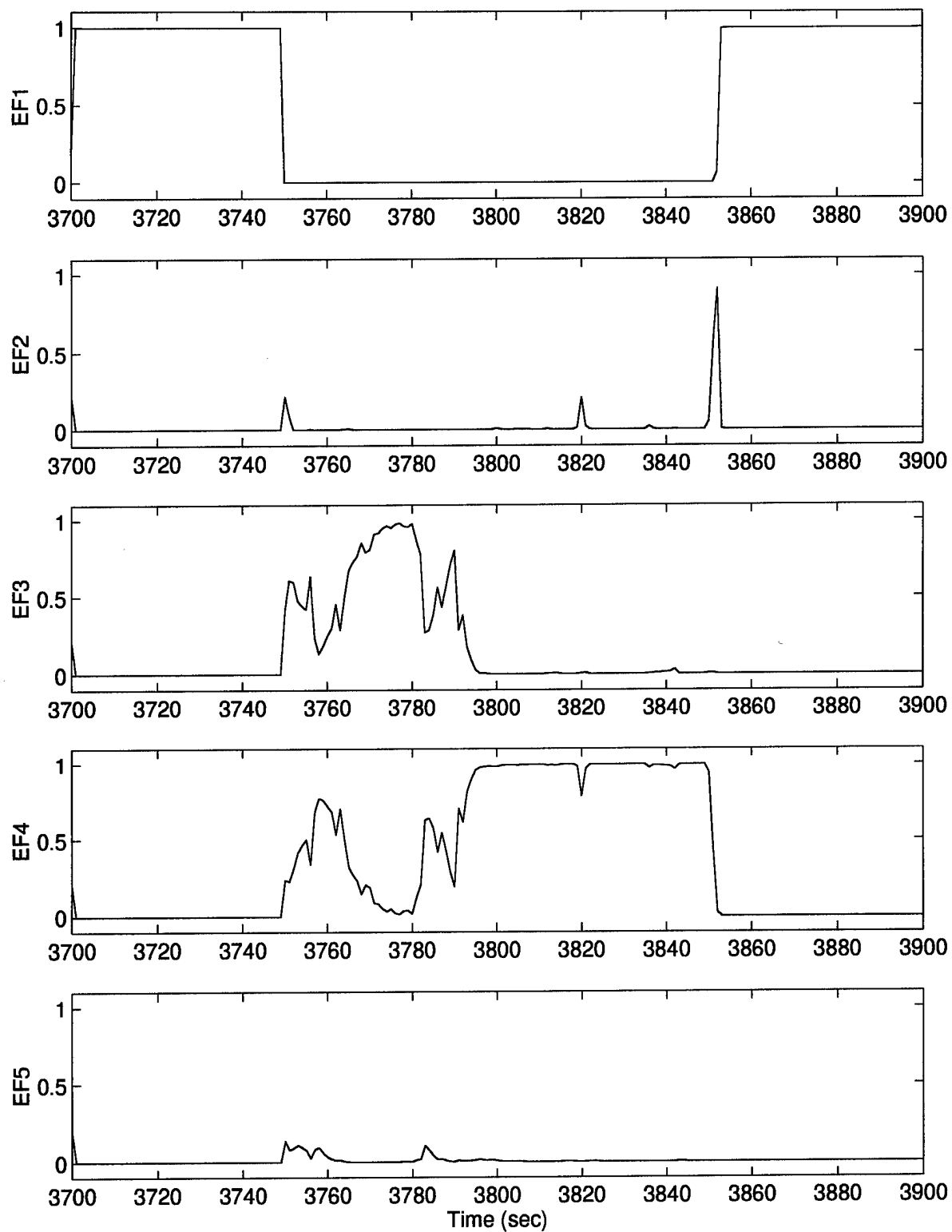


Figure 47. Elemental Filter Probabilities – Case 2: Fixed-Bank

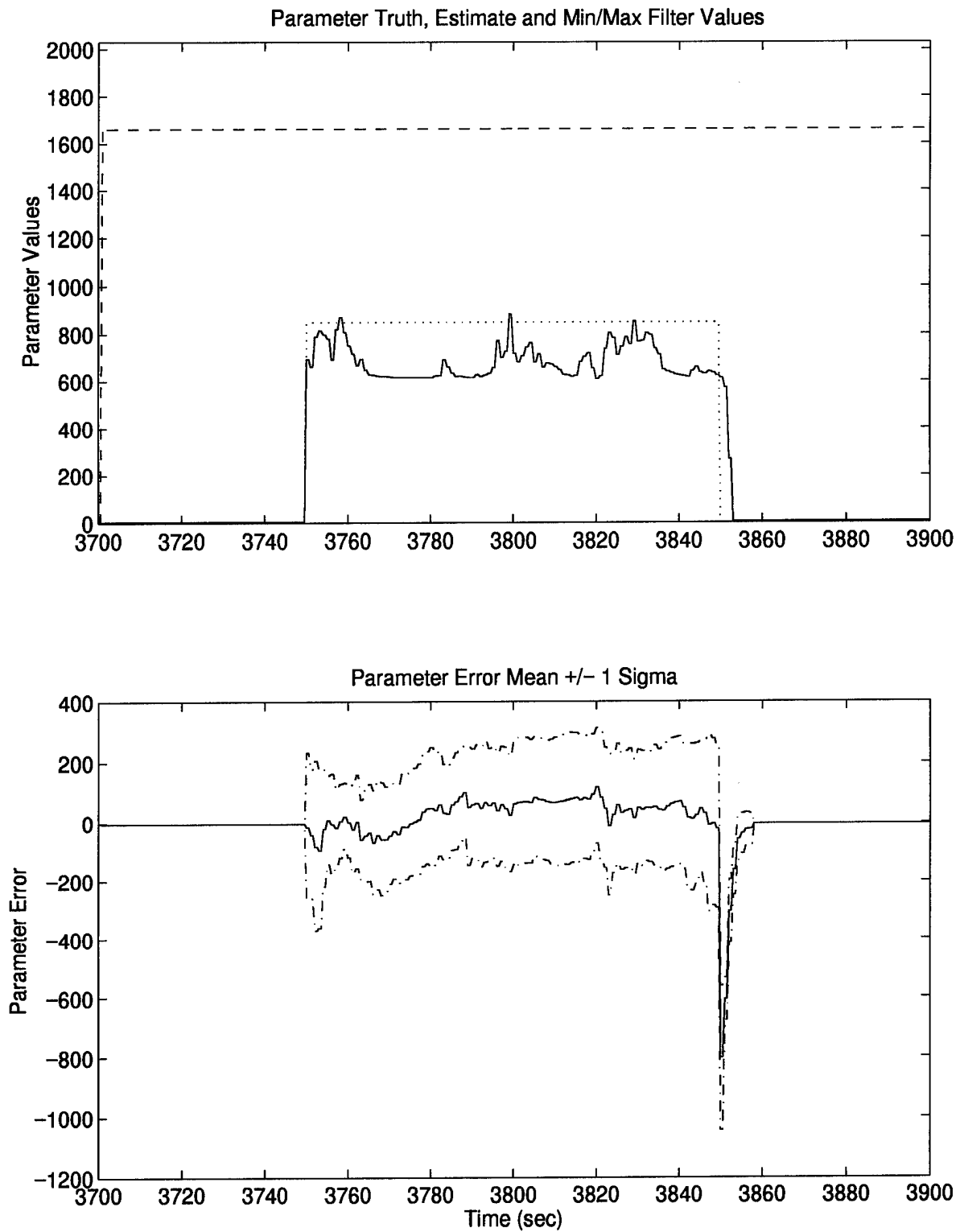


Figure 48. Parameter Estimation Performance – Case 3: Fixed-Bank

is the inability of the fixed-bank MMAE to adapt the parameter choices for the filters such that it could converge on a filter that is a good match to truth. The moving-bank algorithms will exploit their ability to make the necessary adaptation of the parameter choices such that this problem can be avoided.

Case 4: Recall from Table 4 (page 149) that the true parameter undergoes three successive increases in case 4. This case does not provide any new insights beyond those gained from cases 1 – 3 for the fixed-bank approach, but it does help identify potential advantages of utilizing a moving-bank algorithm over using the fixed-bank approach. The parameter estimate shown in Figure 49 reinforces the bias problem encountered with the fixed-bank approach when the true parameter is far from the fixed filter-assumed parameter values. Specifically, an upward bias (above $a_t = 500$) is seen from $t = 3754$ sec to 3800 sec. Similarly, a downward bias (below $a_t = 2000$) is seen from $t = 3850$ sec to 3900 sec, since the largest filter-assumed parameter value is $a_5 = 1660$. The probability and error state estimate plots are shown in Figures 50, 51, and 52 for completeness. The moving-bank algorithms will be discussed in their appropriate sections to identify their potential benefits over the fixed-bank approach for this case study.

Case 5: In this case, the true parameter value takes on the same values as in case 4, but in the opposite sequential order. As with case 4, case 5 is mainly included for completeness and to allow comparison to the density algorithm *with* expansions and increased decision delay discussed presently. Figures 53 – 56 illustrate the fixed-bank performance. Notice the same downward biasing (when $a_t = 2000$) and upward biasing (when $a_t = 500$) that occurred in case 4. Despite this negative tendency, the true parameter value is tracked relatively well once the MMAE algorithm converges to a filter that matches truth reasonably well. For instance, $a_t = 1000$ for $t = 3750$ sec to 3800 sec, and filter 4 ($a_j = 1097$) *finally* possesses the majority of the probability weight at $t = 3790$ sec, resulting in a refinement of the parameter estimate. This refinement is further indicated by the

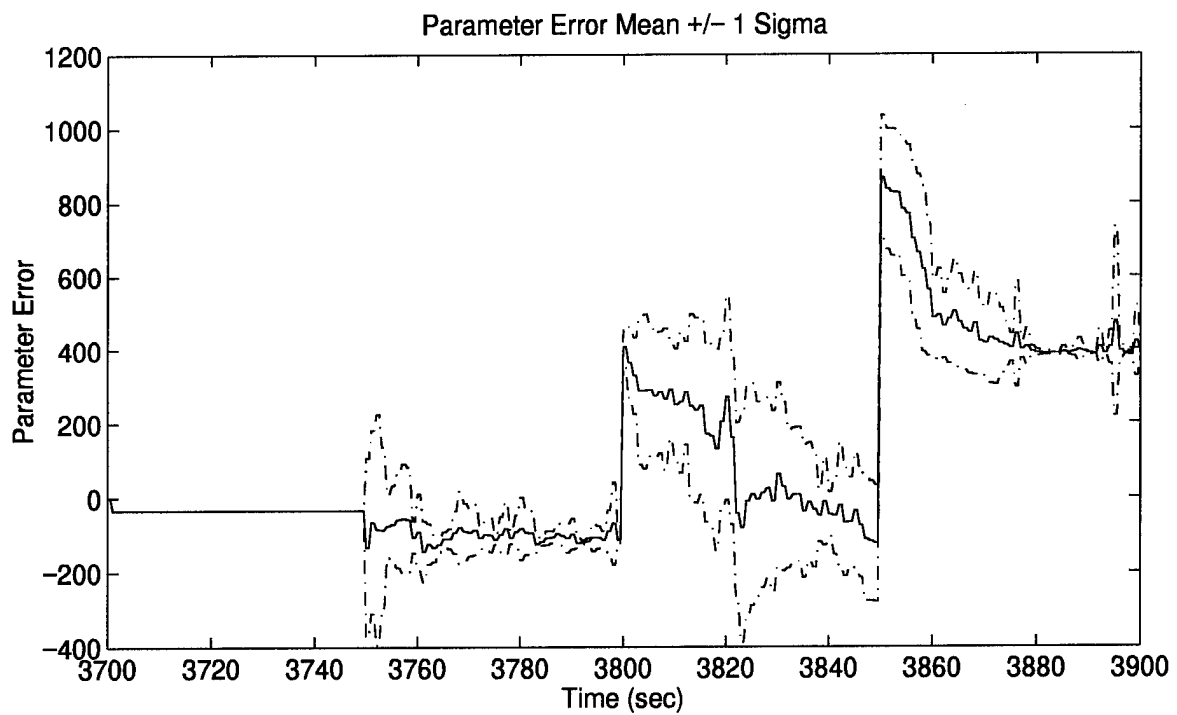
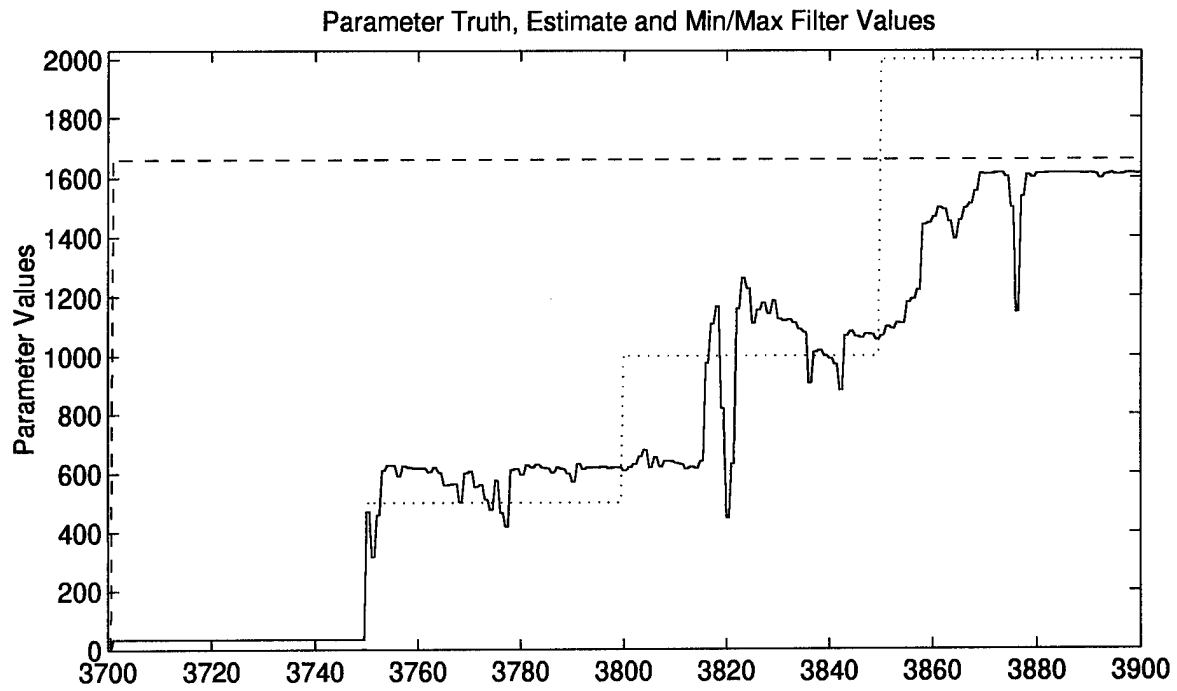


Figure 49. Parameter Estimation Performance – Case 4: Fixed-Bank

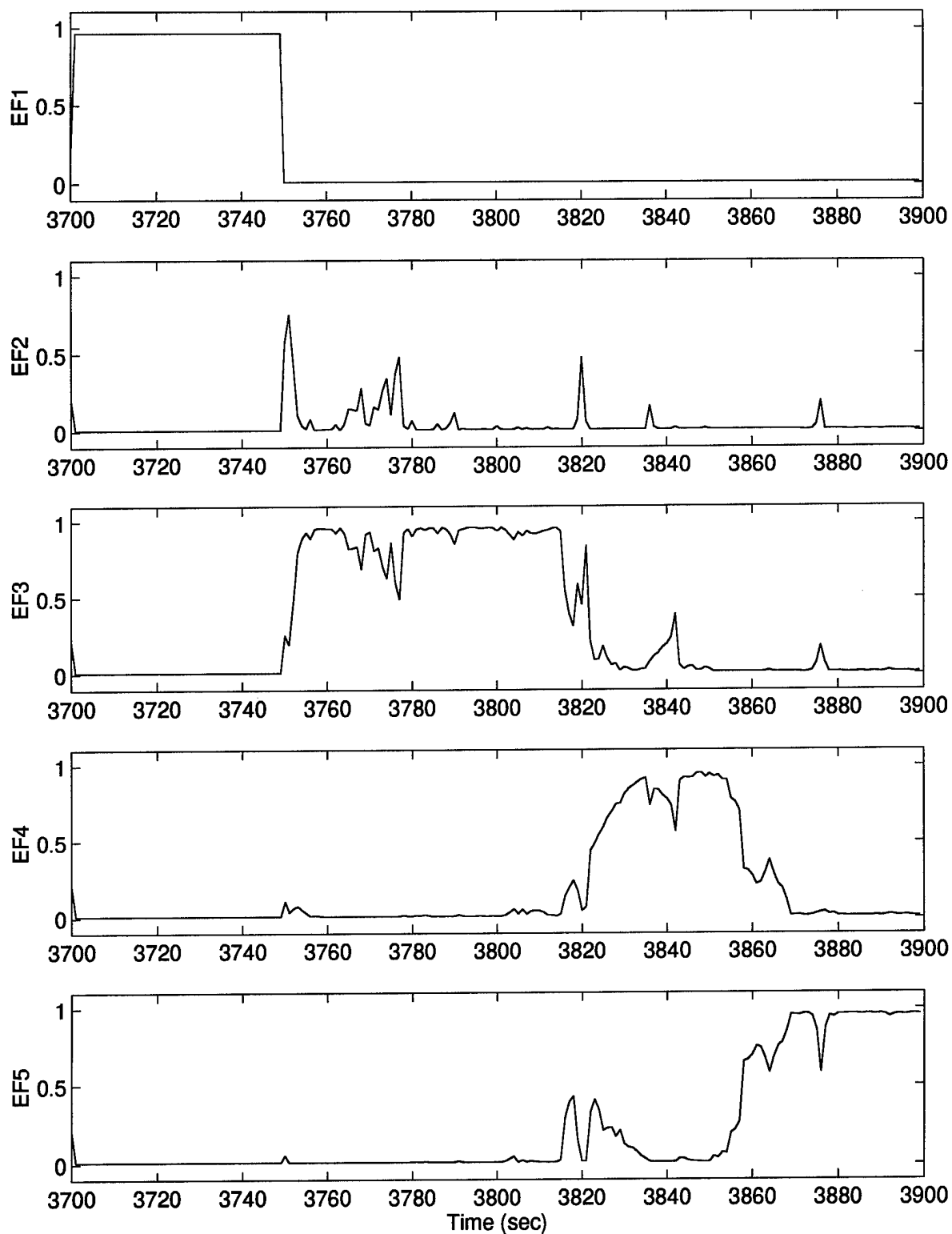


Figure 50. Elemental Filter Probabilities – Case 4: Fixed-Bank

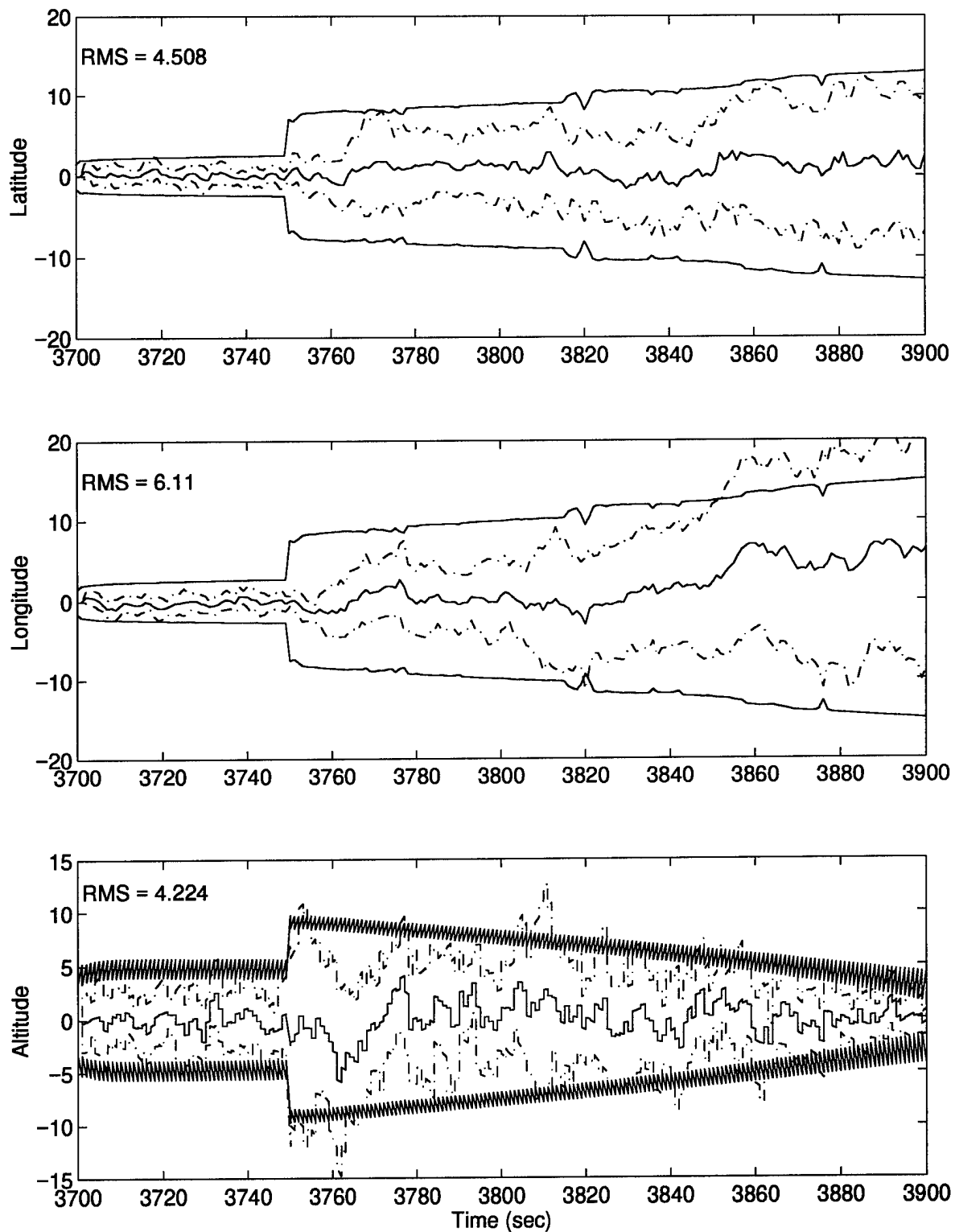


Figure 51. MMAE State Estimation Errors (feet) – Case 4: Fixed-Bank

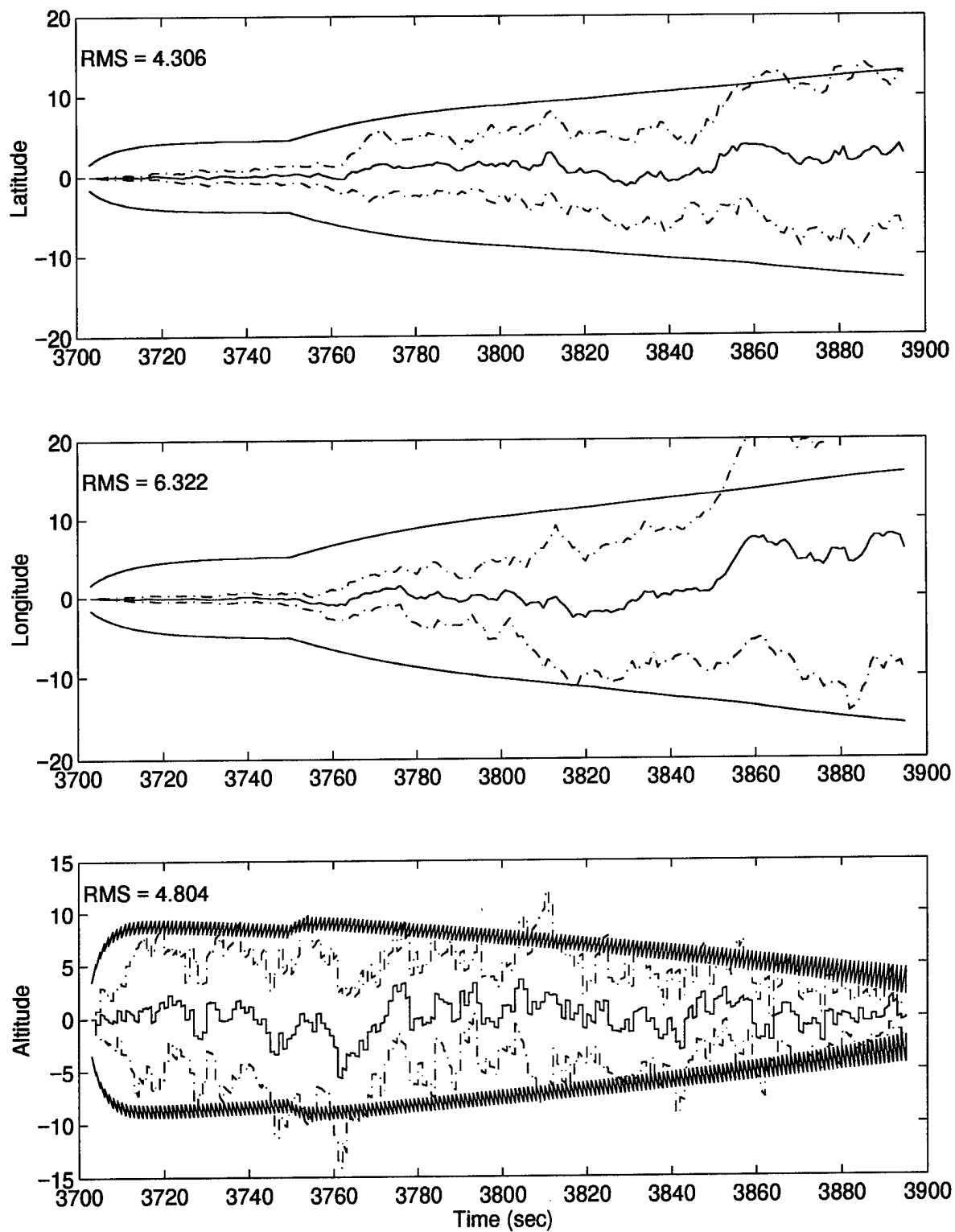


Figure 52. M³AE State Estimation Errors (feet) – Case 4: Fixed-Bank

parameter error mean ± 1 standard deviation plot which shows a near zero mean around $t = 3790$ sec. Similarly, $a_t = 500$ for $t = 3800$ sec to 3850 sec, and filter 3 ($a_j = 615$) *finally* possesses the majority of the probability weight at $t = 3810$ sec. However, since $a_j > a_t$, the upward bias effect is present as stated earlier, and it would be preferable to have greater blending of filters 2 and 3.

4.7 MMAE Incorporating the Density Algorithm

4.7.1 Implementation Issues

The density algorithm presented in Section 3.1 was implemented with five elemental filters. Several design parameters are associated with this algorithm and summarized in Table 6. The design parameters were determined using the guidelines and equations described in Section 3.1. The following paragraphs provide an explanation for these choices.

Table 6. Design Parameters for Density Algorithm

T_1	0.5
T_2	2
T_3	5 (4)
γ	4
T_4	18.5
T_5	0.1
T_6	0.9
# of Foveal Filters	3
Dead1	3
Dead2	4
Initialize New Filters	Yes
Decision Delay	1 (5)
p_{\min}	0.001

Threshold T_1 was found empirically such that a reasonable change in the true parameter value led to a movement decision. For example, if the true parameter, a_t , changed from $a_t = 2000$ to $a_t = 1000$ and the 5 filter-assumed parameter values were $[a_j; j = 1, \dots, 5] = [1066 \ 1300 \ 1533 \ 1766 \ 2000]$, then a move to the left would be desirable. In this case, M_5 changed from 0.29 to 1.15, crossing the threshold $T_1 = 0.5$ and implying that a move should be performed. See the logic given

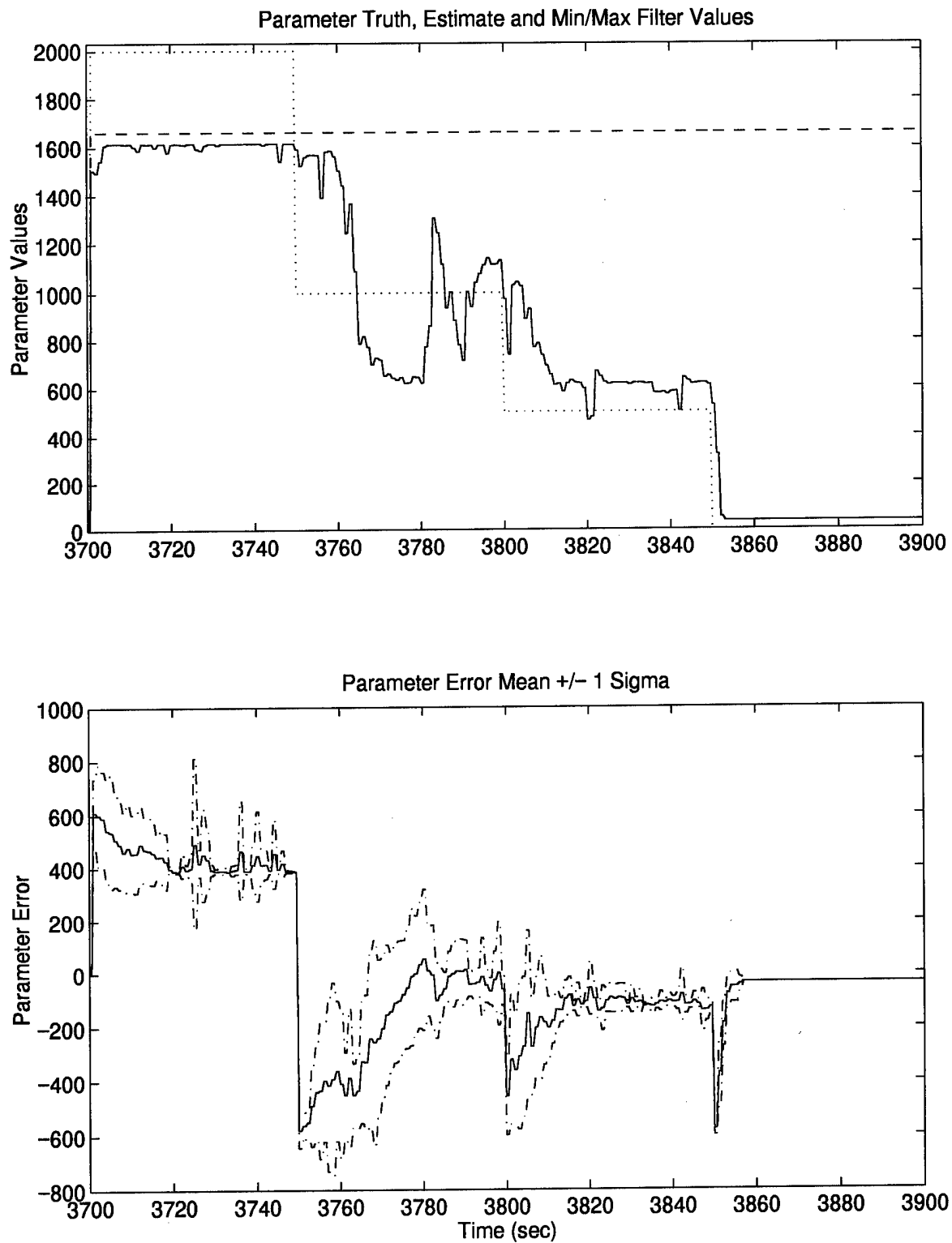


Figure 53. Parameter Estimation Performance – Case 5: Fixed-Bank

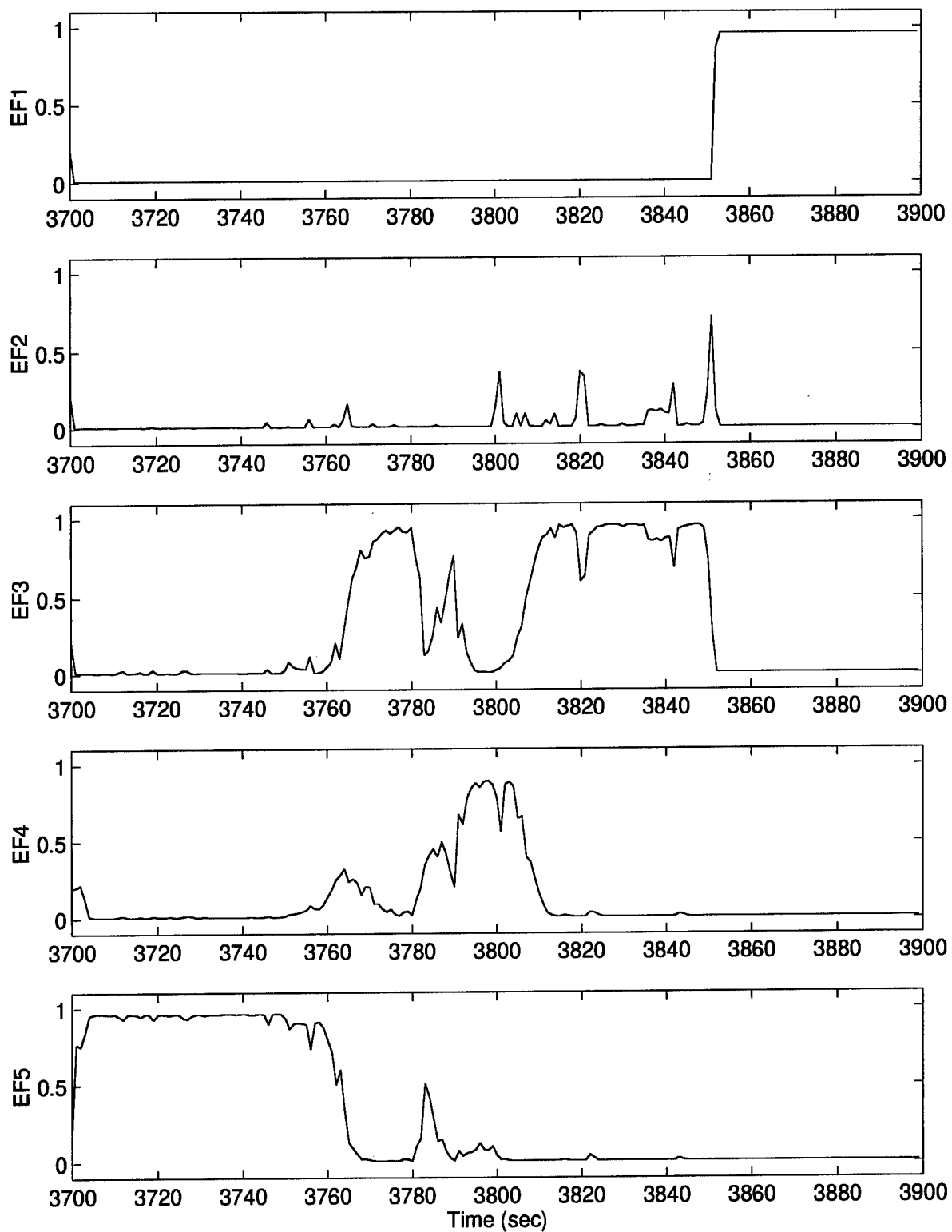


Figure 54. Elemental Filter Probabilities – Case 5: Fixed-Bank

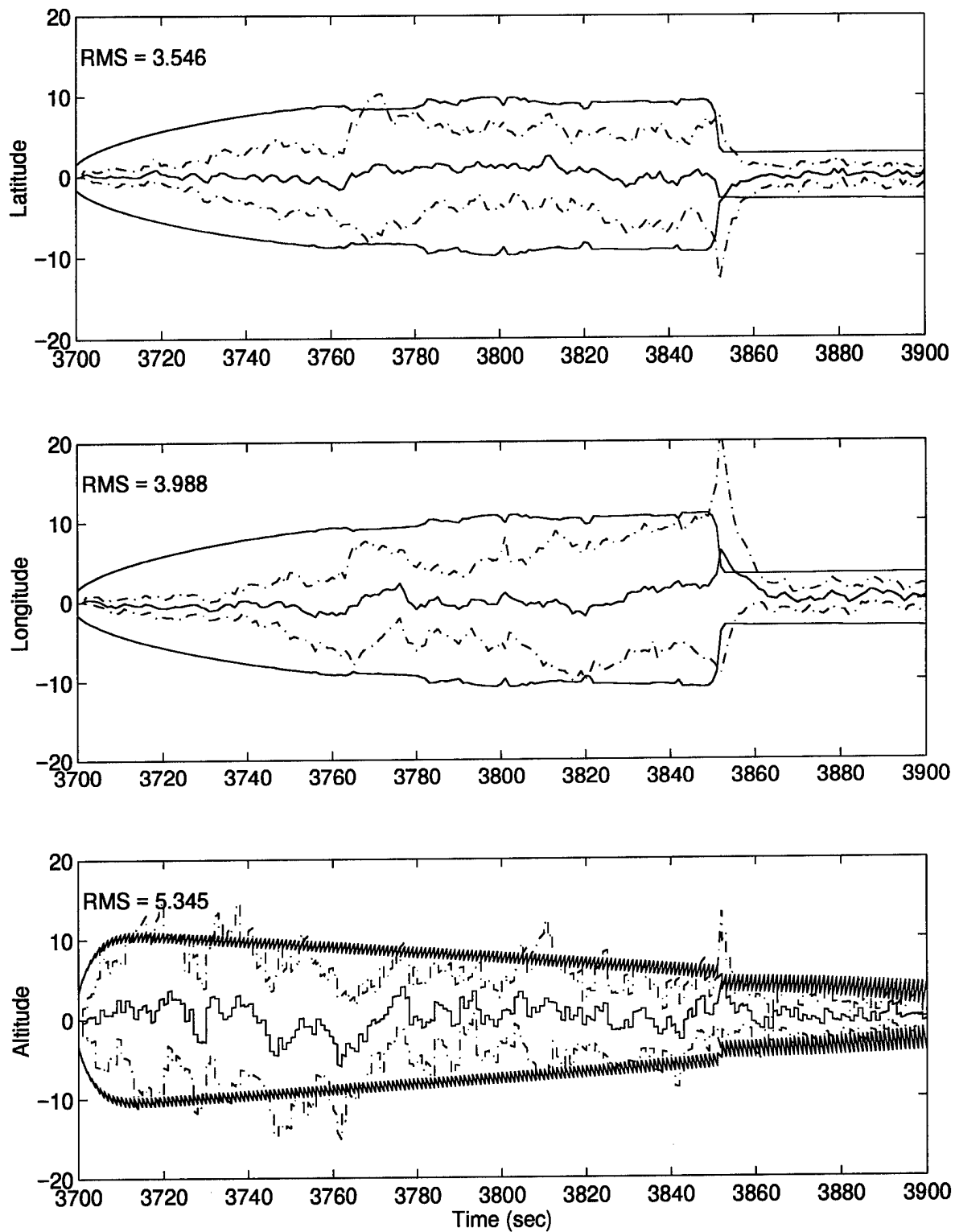


Figure 55. MMAE State Estimation Errors (feet) – Case 5: Fixed-Bank

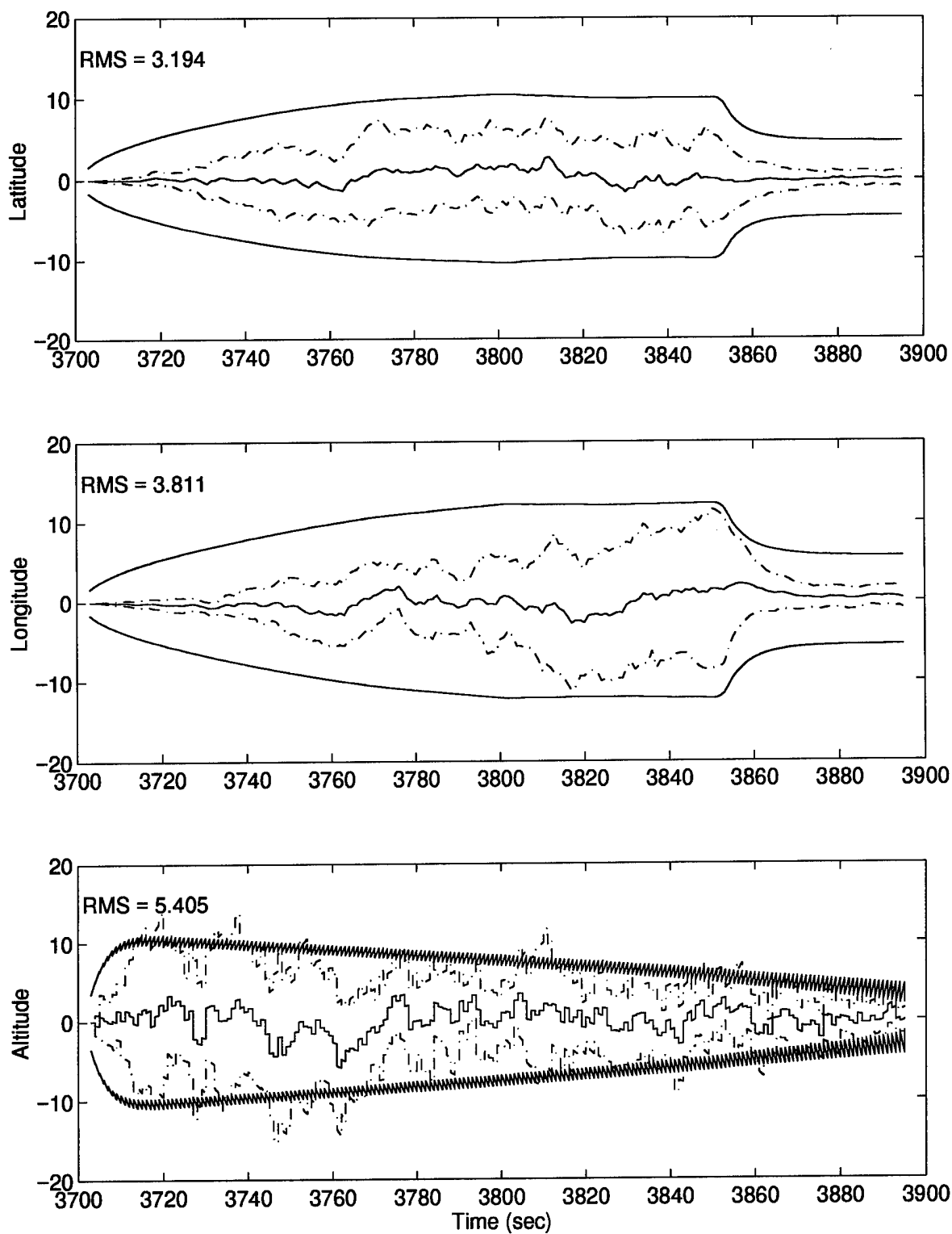


Figure 56. M³AE State Estimation Errors (feet) – Case 5: Fixed-Bank

by Equations (86) and (87). In contrast, if the true parameter changed from $a_t = 200$ to $a_t = 300$ and the 5 filter-assumed parameter values were $[a_j; j = 1, \dots, 5] = [1 \ 188 \ 261 \ 393 \ 523]$, then a move would not be necessary. In fact, M_5 changed from 0.28 to 0.29, which does *not* exceed T_1 , implying that a move should *not* be performed. Repeated analysis of this type for several choices of a_t and a_j led to the selection of $T_1 = 0.5$.

Threshold $T_2 = 2$ implies that a contraction would be executed if only 1 or 2 of the filter-assumed parameter values are close to the best indication of the true parameter value (associated with the highest value of $f_{\mathbf{z}(t_i)|\mathbf{a}, \mathbf{z}(t_{i-1})}$), and the other 3 or 4 filter-assumed parameter values are outliers with respect to truth. See the logic given by Equations (88) and (89).

Note that when $T_3 = 5$, the Decision Delay = 1, and when $T_3 = 4$, the Decision Delay = 5, as indicated by the parentheses and explained presently. The threshold, T_3 , is set to values of 5 and 4 to prevent and allow expansions, respectively. See the logic given by Equations (88) and (90). Threshold $T_3 = 5 = J$ implies that no expansions will be made since the condition $M_6 > T_3$ will never be met. Without the use of expansions, a short decision delay of 1 sample period sufficed to allow transients to die out after a new moving-bank decision had been made. Allowing expansions with the basic density algorithm implementation resulted in erratic changes of the filter-assumed parameter values, and these changes were detrimental to the parameter estimates as presented in the performance section that follows. To illustrate this point, additional simulations were conducted with T_3 set equal to 4, resulting in expansions for some test cases. To help counter the erratic bank changes, a decision delay of 5 sample periods was used. This long delay time induced some sluggishness in the algorithms decision-making process, which is also discussed in the performance section.

The scale factor, γ , was found empirically after choosing T_2 and T_3 such that filters far from the truth, in terms of their parameter values, are not counted via measure M_6 . For example, let

$[a_j; j = 1, \dots, 5] = [1 \ 188 \ 615 \ 1097 \ 1660]$, $a_t = 2000$ and $\gamma = 10$, resulting in $M_6 = 2$ since filters 4 and 5 ($a_j = 1097$ and 1660) are relatively close to truth. A better choice of $\gamma = 4$ leads to $M_6 = 1$ since filter 4 is actually rather far from truth and should not contribute to the summation found in Equation (88) for M_6 . Similarly, let $[a_j; j = 1, \dots, 5] = [1 \ 233 \ 350 \ 525 \ 700]$, $a_t = 500$ and $\gamma = 4$, resulting in $M_6 = 4$, which makes sense since 4 of the 5 filters are close to truth. However, $\gamma = 1.5$ leads to $M_6 = 1$, implying that only 1 filter is close to truth. Although this could be argued as a correct assessment, letting $\gamma = 1.5$ versus $\gamma = 4$ would lead to unneeded contractions based on the logic in Equation (89) and potentially eliminate the elemental filter's residual distinguishability. In other words, the bank could contract to the point that all the filter models would look equally good, and parameter estimation would tend to degrade. This type of analysis led to the selection of $\gamma = 4$.

Threshold $T_4 = qchisq(P_x, m)$ was calculated via MATHCAD [39] such that the probability of correctly classifying good filters was 99.5% with 6 measurements sources as shown below

$$T_4 = qchisq(0.995, 6) = 18.5$$

This paragraph will discuss the selection of thresholds T_5 and T_6 . Threshold T_6 was found empirically such that a medium change in the true parameter value led to a medium movement decision. For example, with a_t changing from 1000 to 500 and filter-assumed parameter values of $[a_j; j = 1, \dots, 5] = [1000 \ 1300 \ 1533 \ 1766 \ 2000]$, a threshold of $T_6 = 0.9$ led to $M_9 = 5 = J$, resulting in a medium move. See the logic given by Equations (97) and (104). After this medium move, M_9 reduced to 3 or 4 (negating the possibility of another medium move) since the ratio

$$\frac{\mathbf{r}_{M_4}^T \mathbf{A}_{M_4}^{-1} \mathbf{r}_{M_4}}{\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j}$$

used in calculating M_9 fell outside the range

$$1 - T_6 \leq \frac{\mathbf{r}_{M_4}^T \mathbf{A}_{M_4}^{-1} \mathbf{r}_{M_4}}{\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j} \leq 1 + T_6$$

for one or two of the filters. Specifically, the new parameter values assigned after the move, $[a_j; j = 1, \dots, 5] = [1 \ 233 \ 350 \ 525 \ 700]$, allowed one or two filters to appear “moderately similar” in terms of their likelihood quotients. Notice that $a_t = 500$ is well within the parameter span of the MMAE and additional medium moves are not necessary at this time. The same process discussed here for selecting T_6 led to selecting T_5 such that a large change in the true parameter value led to a hard movement decision.

The number of foveal view filters is set equal to 3 and is motivated by having 3 out of 5 filters focused on the true parameter value and 2 out of 5 filters positioned as outliers, maintaining a peripheral view of the parameter space. This is based on the design approach used in this simulated example problem. Recall that the contraction and expansion factors are given by

$$\text{Contraction Factor} : \kappa = \# \text{ of desired counted (foveal) filters}$$

and

$$\text{Expansion Factor} : \varepsilon = \frac{M_6}{\# \text{ of desired counted (foveal) filters}}$$

The dead zone region was empirically determined to prevent unneeded soft moves when the filter associated with the maximum probability is close to the center of the bank. Notice that the dead zone covers the region defined by filter numbers 3 – 4; so a soft move is prevented if either filter 3 or 4 is associated with the maximum probability weight. A single filter dead zone located at filter 3 was found to be ineffective at preventing many unwanted movements, so larger dead zones were considered. Initially, a dead zone spanning filters 2 – 4, which is symmetric about the center of the bank (and is thus conceptually appealing at first), was proposed but found to prevent desired moves to the left. This raises a problem-dependent characteristic of an MMAE for which the measurement

noise covariance matrix, \mathbf{R} , is the uncertain parameter. With \mathbf{R} as the parameter being estimated, the probability calculation within the MMAE will tend to favor elemental filters having conservative (larger) values of \mathbf{R} (see discussion in Section 3.1.5.2). Therefore, the dead zone defined by filters 3 – 4 was more effective than a dead zone of 2 – 4, since it accounted for this attribute of the MMAE.

Filters taking on a new parameter value as a result of a moving-bank decision were newly initialized with $\hat{\mathbf{x}}_{\text{MMAE}}$. Finally, a lower bound on the probabilities of $p_{\min} = 0.001$ was used for consistency.

4.7.2 Performance *Without* Expansion and Additional Delay

Case 1 : The adaptive nature of this algorithm is illustrated by the parameter estimate performance shown in Figure 57. Note that the two traces (\cdots) and ($---$) are overlaid from $t = 3755$ sec to 3850 sec, causing what appears to be a trace ($\cdot - \cdot$). In reality, the maximum filter-assumed parameter value indicated by ($---$) is equal to the true parameter value indicated by (\cdots).

Filters 4 and 5 have the strongest influence on the parameter estimate from $t = 3750$ sec to 3850 sec, as realized from the probability plot in Figure 58, while filters 1 – 3 have probabilities at or near the lower bound of $p_{\min} = 0.001$. As a result, the parameter estimate is seen to increase quickly (within 5 sample periods) to a value of $\hat{\mathbf{a}}_{\text{MMAE}} = 1494$, which is in the neighborhood of truth, $\mathbf{a}_t = 2000$. Notice that, up to this point, the parameter estimate in Figure 57 is the same as the parameter estimate in Figure 40. More importantly, the density algorithm invokes a soft move to the right at $t = 3755$ sec (i.e., upward on Figure 57), allowing filter 5 to assume the value $\mathbf{a}_5 = 2000$ and an increase in $\hat{\mathbf{a}}_{\text{MMAE}}$ to 1878 in *one* sample period. Filter 1 also assumes a larger parameter value at this time, and this bank move is indicated by the trace pair ($---$) which shows an increase in both the minimum and maximum filter-assumed parameter values, i.e., a move right. At $t = 3782$ sec, 3784 sec, and 3823 sec, the algorithm invokes contractions, resulting in further increases

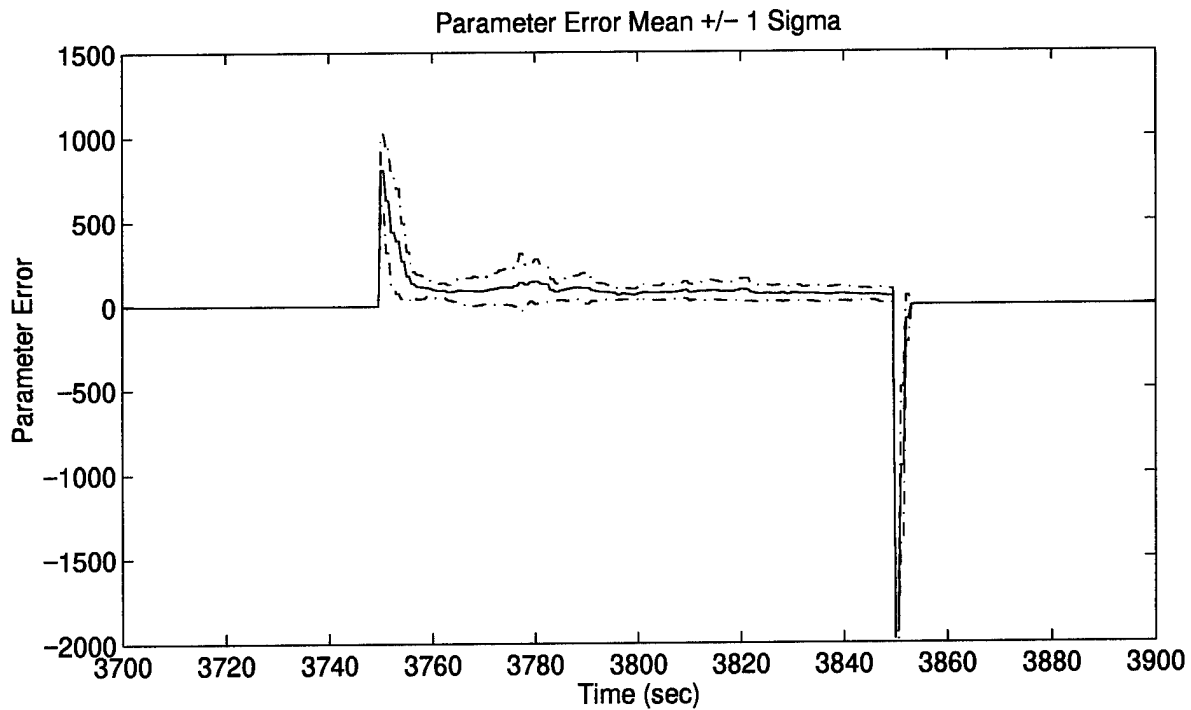
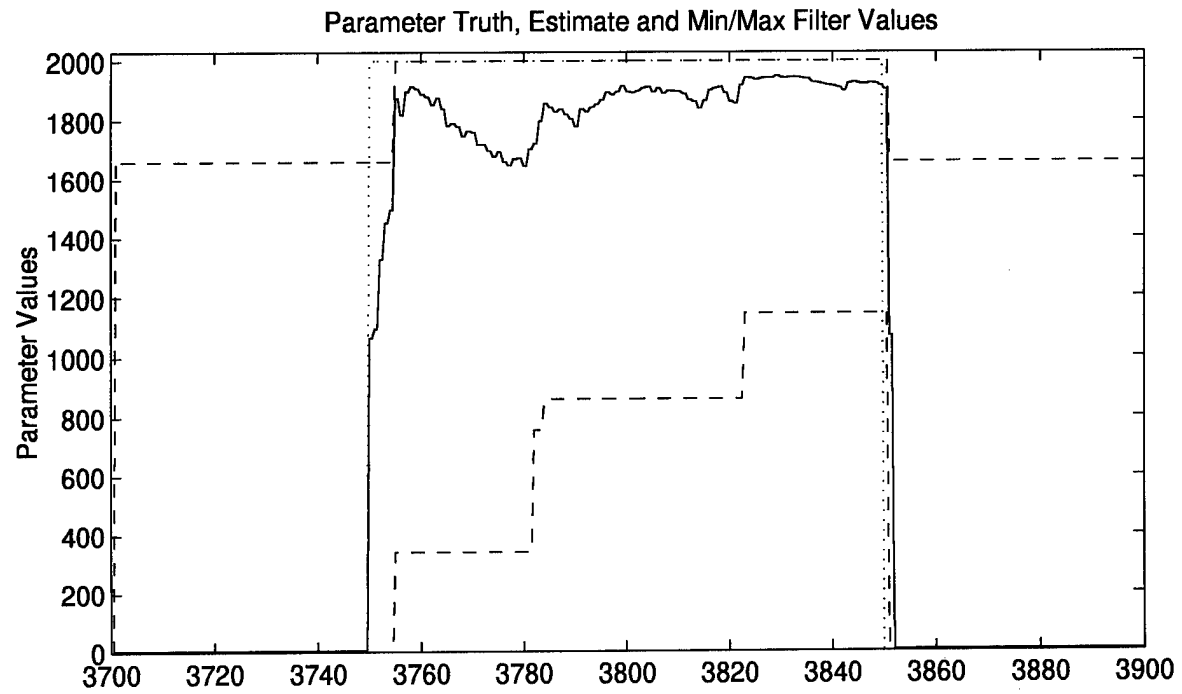


Figure 57. Parameter Estimation Performance – Case 1: Density Algorithm

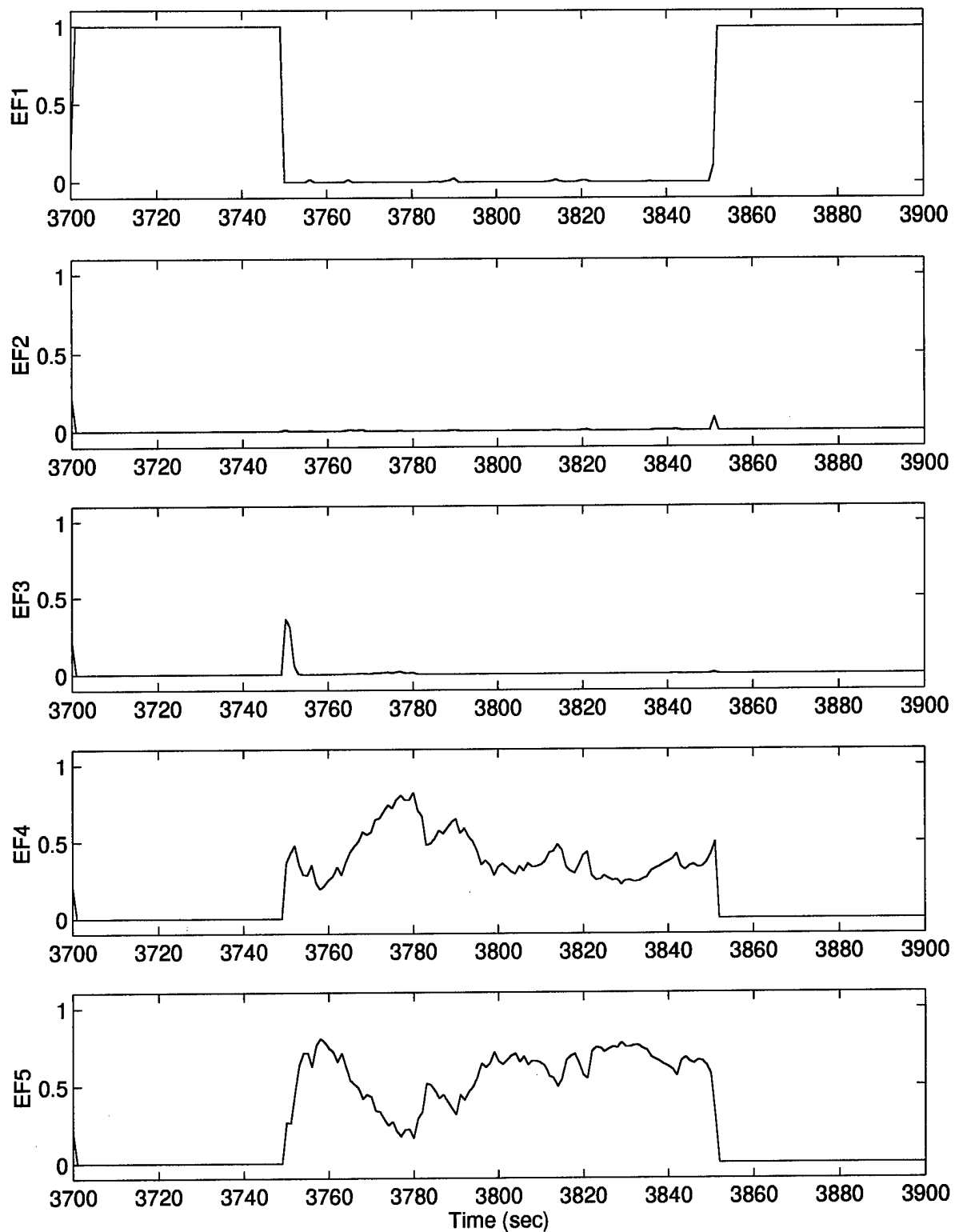


Figure 58. Elemental Filter Probabilities – Case 1: Density Algorithm

of the assumed parameter values for filters 1 – 4 and a refinement of the parameter estimate. The MMAE blending of $a_4 = 1786$ and $a_5 = 2000$ still causes the parameter estimate to be biased below $a_t = 2000$, but this becomes less significant as the bank is repeatedly contracted. The parameter estimation measure, \hat{e}_{RMS}^a , is shown in Table 16 (Appendix E) as being one of the most competitive among the algorithms.

The MMAE blended and M^3 AE final state estimation errors are shown in Figures 59 and 60. The altitude state is well-tuned for both the MMAE and M^3 AE error state estimates with some conservatism prior to the onset of interference/jamming ($t < 3750$ sec) for the M^3 AE error state estimate. Again, this is due to the blended parameter estimate being biased high in the unjammed environment, as explained under case 1 for the fixed-bank MMAE. The primary difference between the results shown here and those for the fixed-bank MMAE is the lack of large transients at $t = 3850$ sec for the MMAE estimates, resulting in cleaner transitions for the M^3 AE estimates. Recall the case 1 fixed-bank discussion which identified the excessive time (5 sample periods) needed by the MMAE to converge on the true parameter value when the interference/jamming is removed at $t = 3850$ sec. In contrast, the density algorithm converges in 3 sample periods, which is evident by observing the raw probability data (not shown here) along with the parameter estimate plot in Figure 57. This improvement in the conventional MMAE's blended estimates is passed on to the final M^3 AE estimates, resulting in better adaptive tuning to the actual measurement noise variances, and smaller standard deviations for the error state estimates from $t = 3850$ sec to 3870 sec.

Case 2: The parameter estimate plot in Figure 61 shows that, following a contraction at $t = 3757$ sec and some transient behavior in the probabilities shown in Figure 62, the parameter estimate tracks truth relatively well. The state plots are not included as they look extremely similar to those in case 1, and the performance measures \hat{e}_{RMS}^a and \hat{e}_{RMS}^x (see Tables 16 – 22 in Appendix E) are more useful.

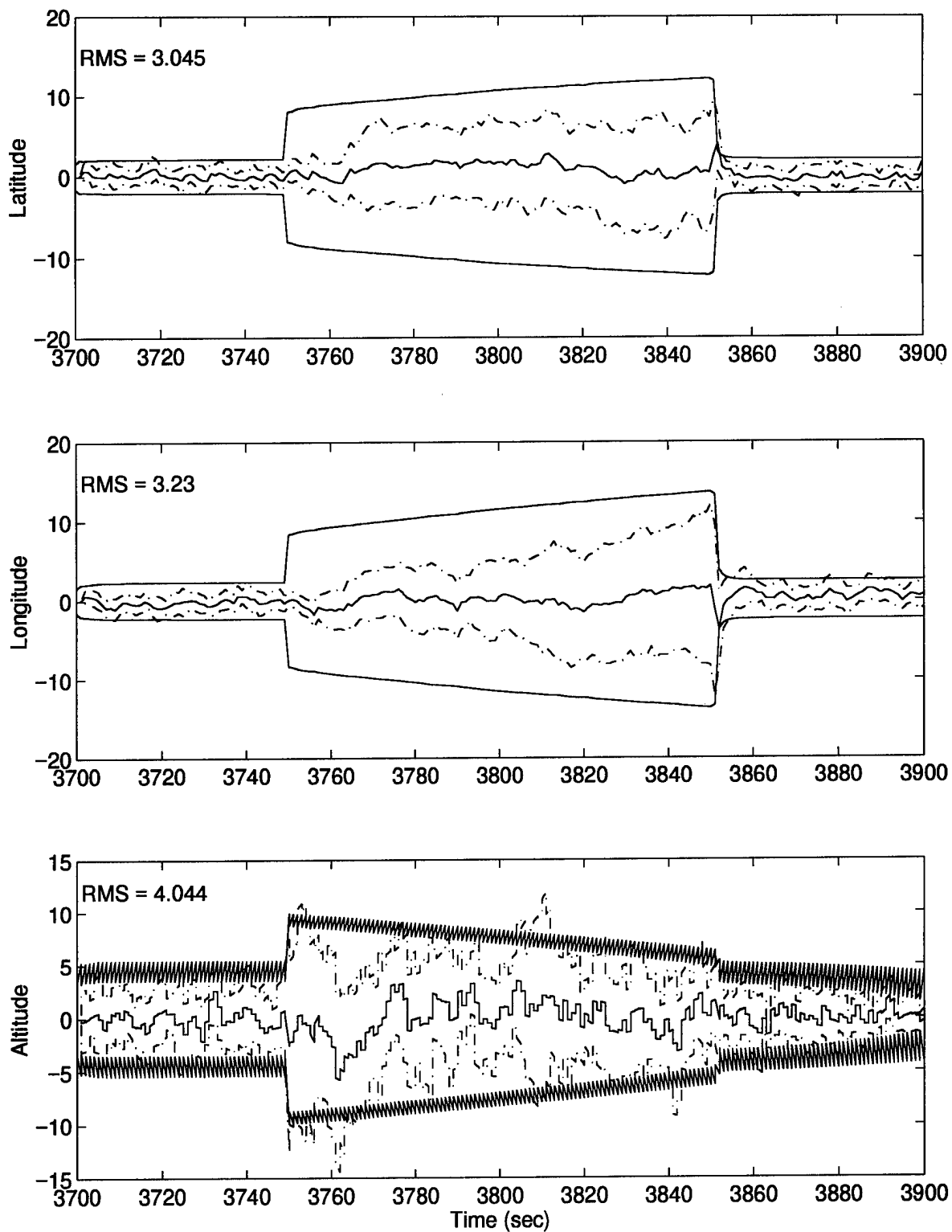


Figure 59. MMAE State Estimation Errors (feet) – Case 1: Density Algorithm

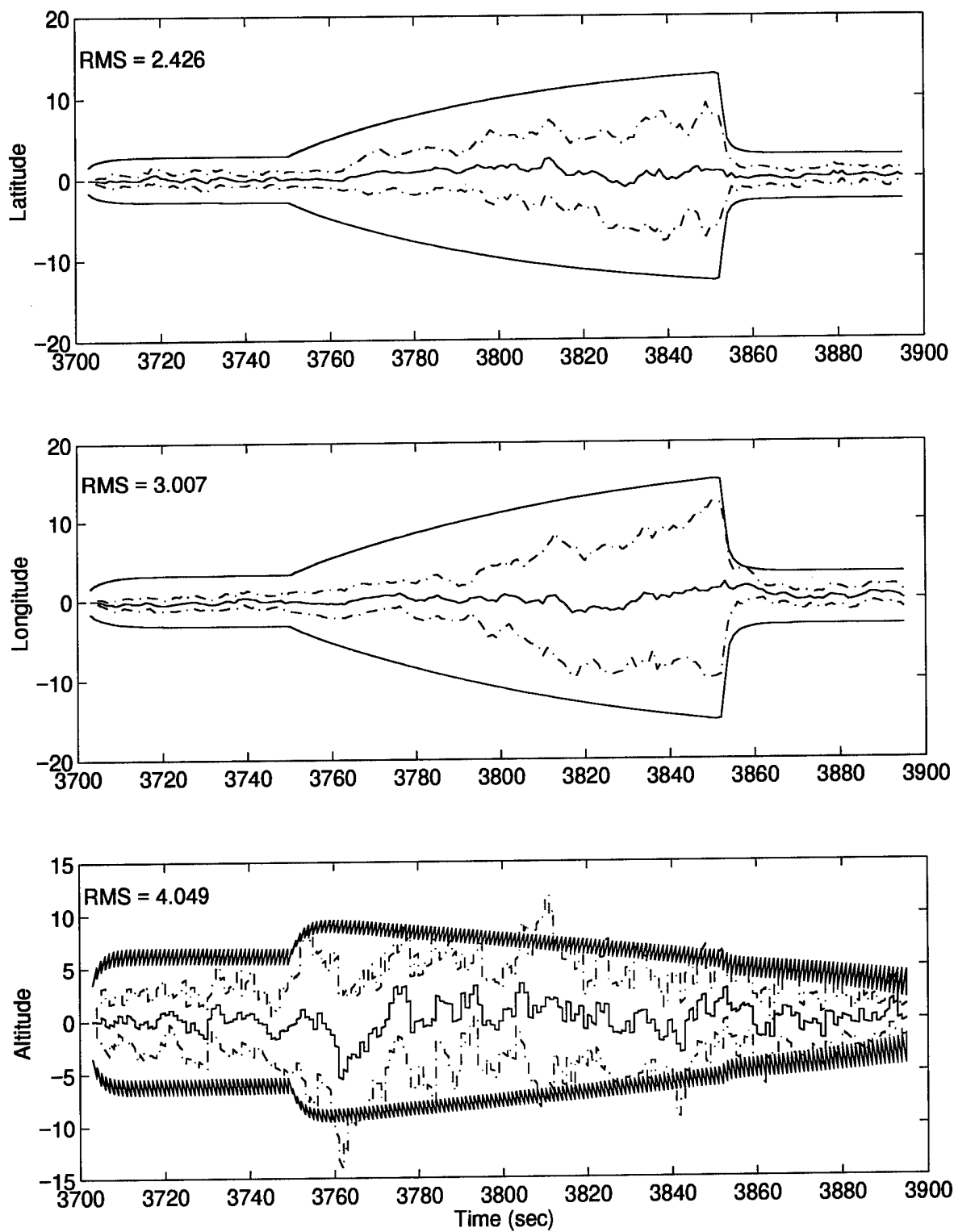


Figure 60. M³AE State Estimation Errors (feet) – Case 1: Density Algorithm

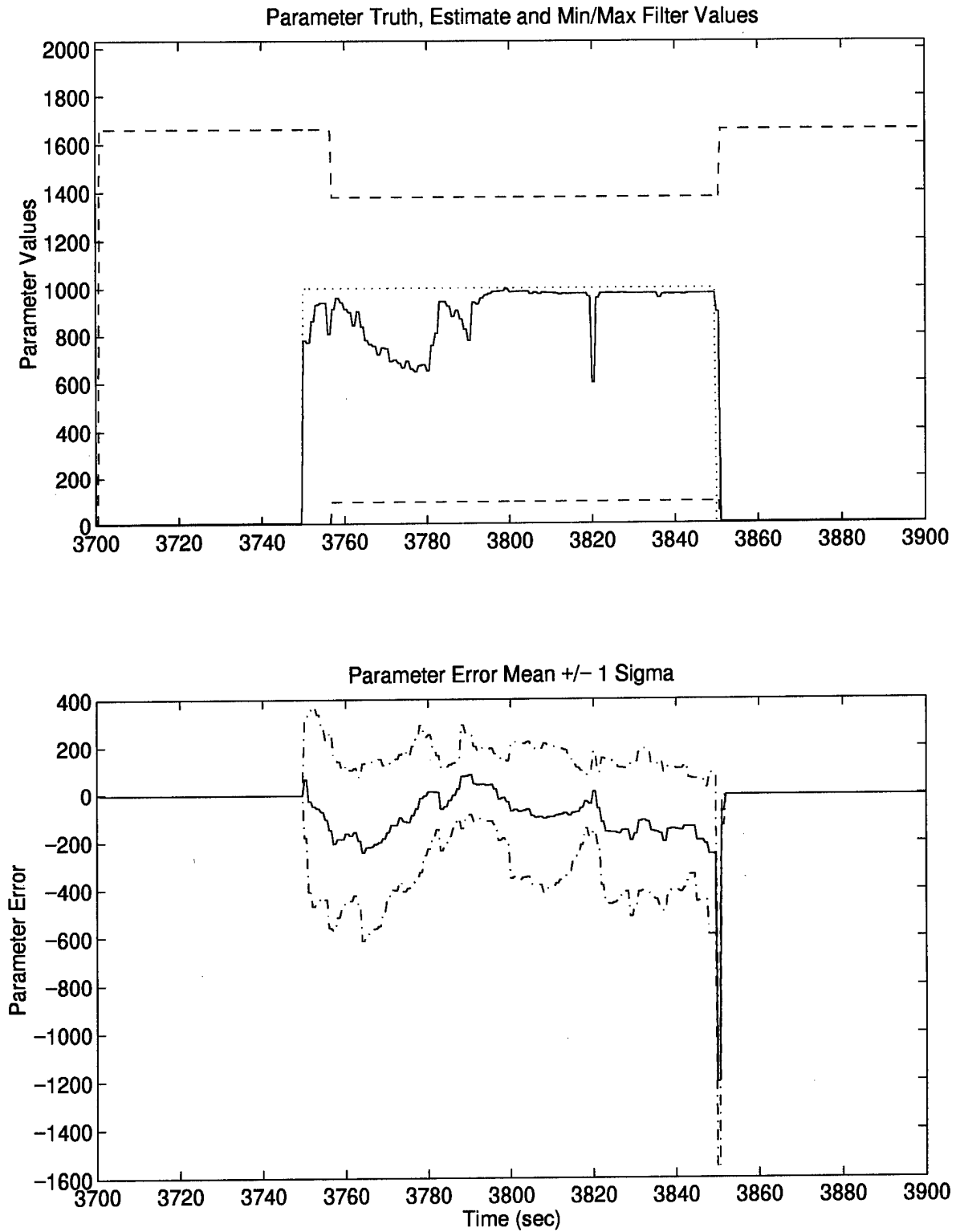


Figure 61. Parameter Estimation Performance – Case 2: Density Algorithm

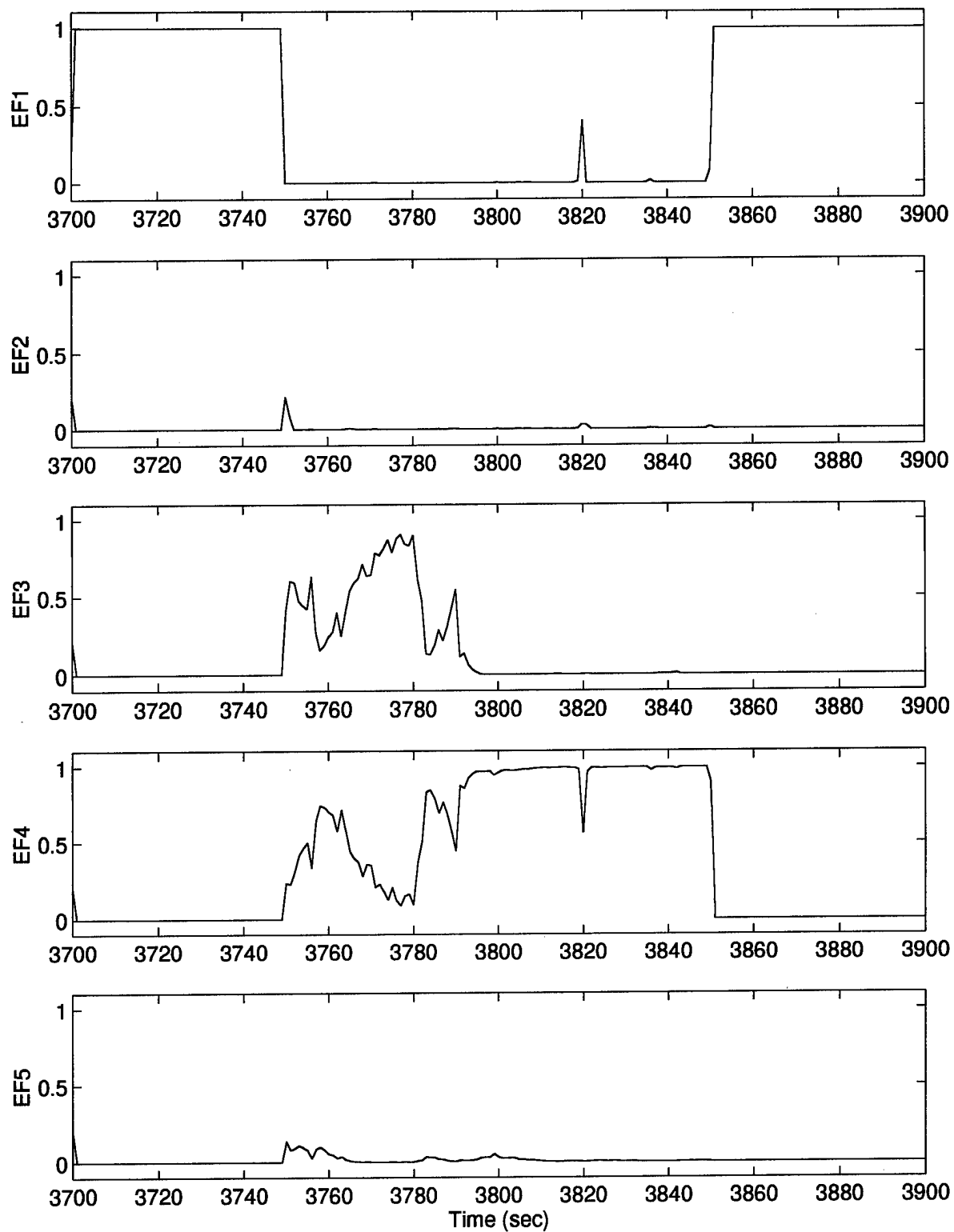


Figure 62. Elemental Filter Probabilities – Case 2: Density Algorithm

Case 3 : The plots for this case are virtually identical to those for case 2 with some improvement in the performance measures, and so they are not presented. Even more significant is the fact that the fixed-bank performance degrades between cases 2 and 3, whereas the density algorithm performance does not. Recall the discussion presented earlier for the fixed-bank performance which identified that this degradation results from the true parameter value lying between two filter-assumed parameter values. The ability of the density algorithm to adapt its filter-assumed parameter values accounts for the improved parameter estimation measure between cases 2 and 3, i.e., \hat{e}_{RMS}^a improves from 146 to 110. In contrast, the fixed-bank value for \hat{e}_{RMS}^a increases from 71 to 128. As expected, \hat{e}_{RMS}^a associated with the density algorithm is better than \hat{e}_{RMS}^a associated with the fixed-bank for case 3, i.e., $110 < 128$.

Case 4 : Some benefits of this moving-bank algorithm versus the fixed-bank approach are highlighted in this case study. The density algorithm provides a vehicle to modify the filter-assumed parameter values such that one filter is a particularly good match to truth. This is seen by comparing the parameter estimation performance plots shown in Figures 49 and 63 for these two algorithms. In particular, notice the bank motion illustrated by the trace pair (- - -), and the density algorithm's ability to "surround" the true parameter value with a bank that has contracted about that true parameter value. The two time frames when the fixed-bank suffers a bias on the estimate are from $t = 3754$ sec to 3800 sec and again for $t = 3850$ sec to 3900 sec. A series of two contractions by the density algorithm at $t = 3751$ sec and 3757 sec allows the moving-bank MMAE to provide a refined estimate of the parameter as compared to the fixed-bank MMAE. Similarly, a hard move right at $t = 3845$ sec followed by a soft move right at $t = 3852$ sec brings the filter-assumed parameter value of filter 5 close to truth, and this filter obtains the majority of the probability weight, as seen in Figure 64. However, the hard move right is partially made necessary by an undesirable medium move left at $t = 3843$ sec. This is due to unusually large (or small) noise samples at each update time for the

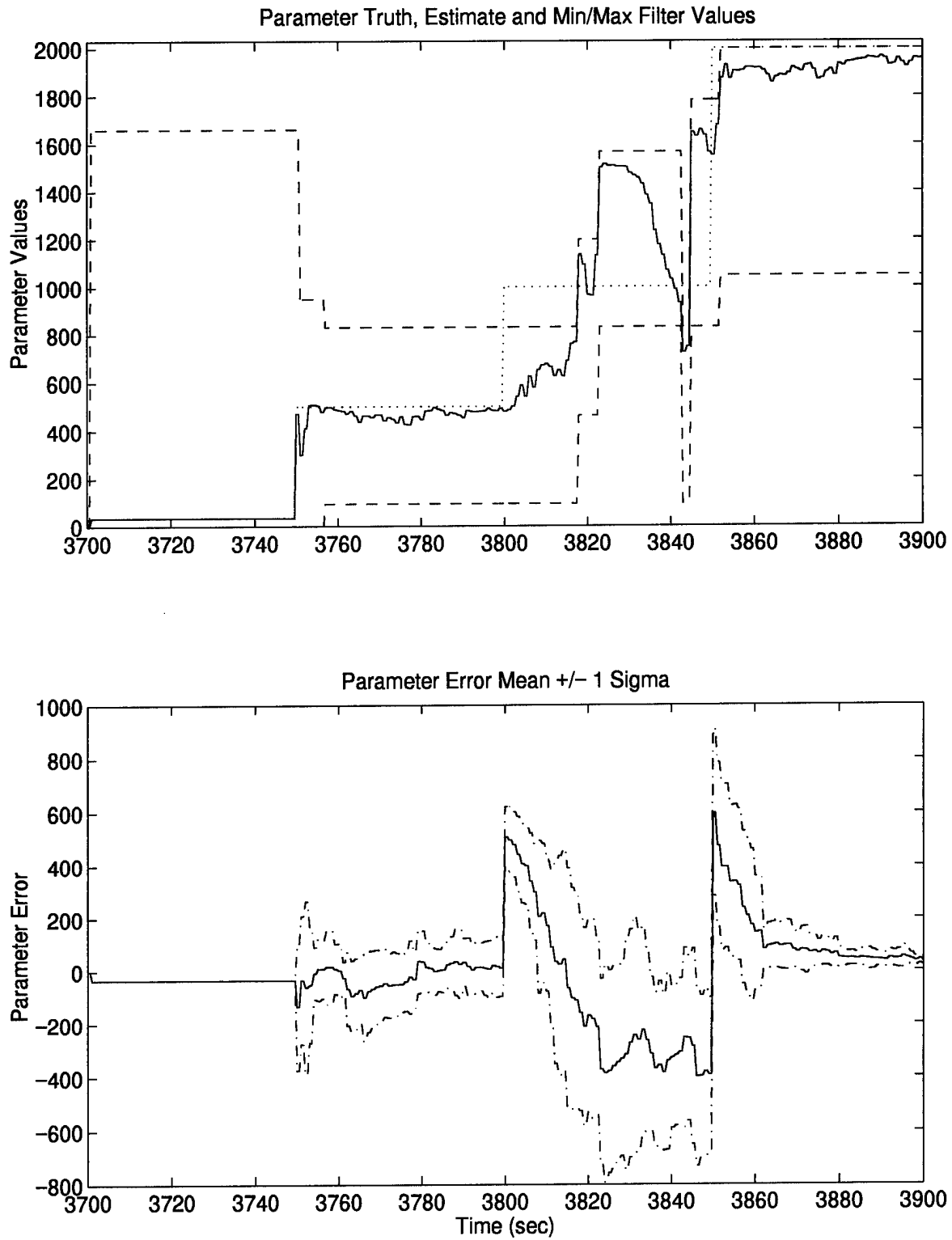


Figure 63. Parameter Estimation Performance – Case 4: Density Algorithm

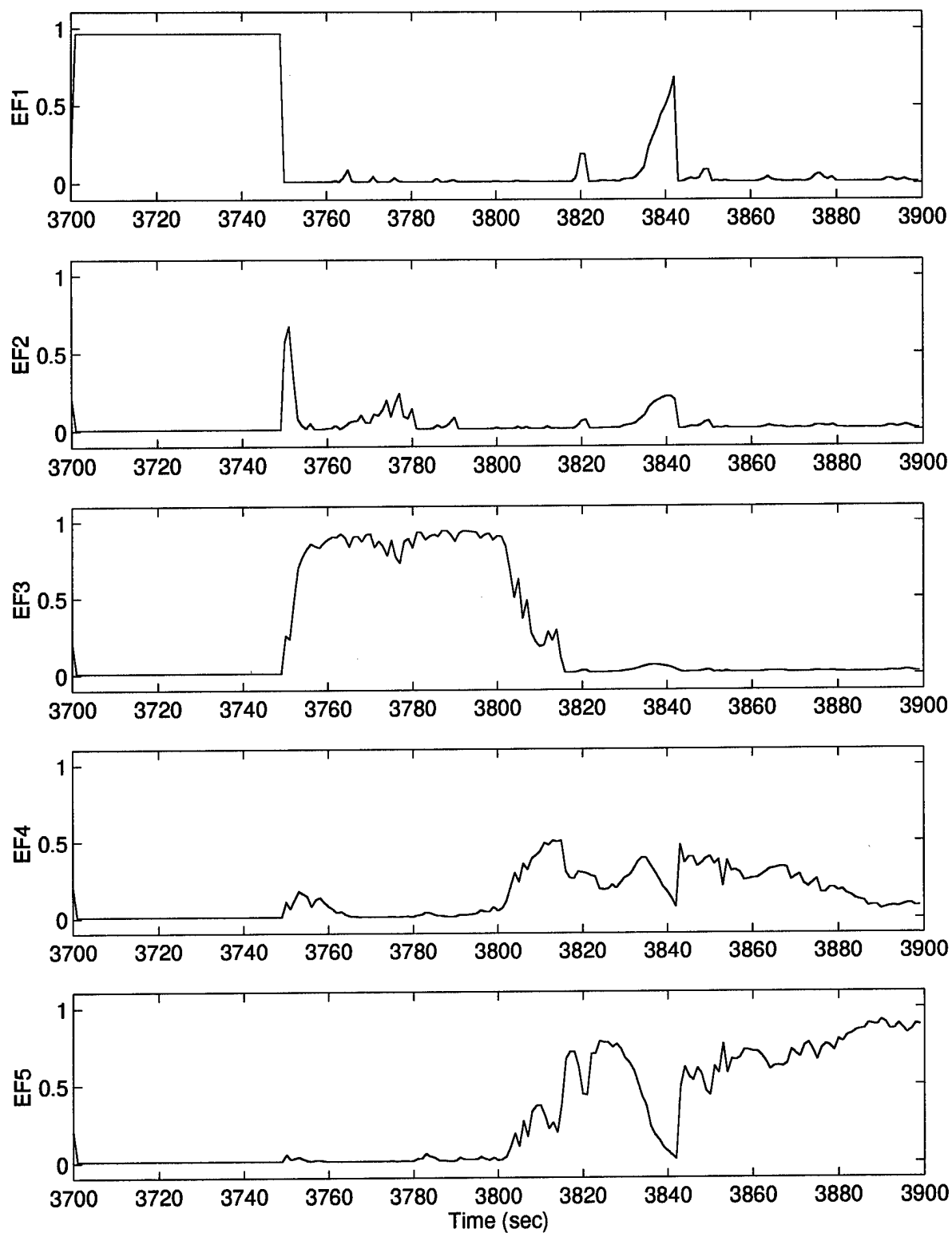


Figure 64. Elemental Filter Probabilities – Case 4: Density Algorithm

particular stochastic process sample used for this simulation run, which induced the medium move left and identifies a potential liability in the density algorithm. Invoking a decision delay will help remove oscillatory bank movement (i.e., move left then right then left ...), but this will not overcome unusually larger (or small) noise samples. An attempt was made to smooth out the effect of such noise samples by averaging data samples of the density $f_{z|a,z}$ over 2 to 10 samples. Unfortunately, the densities for a given filter affected by these unusually large noise samples could vary by several orders of magnitude, requiring extreme smoothing which then resulted in unacceptable decision delays for cases in which these noise samples were not a problem. These unusual noise samples are problematic for all the case studies at $t = 3820$ sec, since the same random seeds which initialize the random number generator were used for each case.

Despite the density algorithm's ability to avoid some of the bias effects plaguing the fixed-bank MMAE, some biasing is encountered in the regions of $t = 3800$ sec to 3817 sec and 3823 sec to 3835 sec. This motivates a recommendation in Chapter 5 to pursue a methodology which estimates or somehow accounts for this biasing. The discussion in Section 3.1.5.1 focused on the residual covariance matrix given by

$$\mathbf{A}_j = \mathbf{H}_j \mathbf{P}_j^- \mathbf{H}_j^T + \mathbf{R}_j$$

and identified that large values of R_j in the scalar measurement case or large eigenvalues of \mathbf{R}_j with $m > 1$, resulted in more conservative tuning. The discussion then described the MMAE's tendency to favor more conservatively tuned filters. The explanation for this tendency was clear for cases where the measurement noise, \mathbf{R}_j , is the uncertain parameter. However, the dynamics driving noise covariance, \mathbf{Q}_{dj} , also determines how conservatively tuned the filters are and may produce the same biasing effect. Furthermore, uncertainties in the output matrix, \mathbf{H}_j , or the state transition matrix, Φ_j , could produce this same biasing trend as they effect the calculation of \mathbf{A}_j shown above either directly in the case of \mathbf{H}_j or indirectly through \mathbf{P}_j in the case of Φ_j (see Equation (4) on page 14).

Finally, it is *not* anticipated that uncertainties in the control input matrix, B_{dj} , will produce this same biasing effect. Observation of Equations (3) – (8) reveals that B_{dj} simply impacts the state estimates, $\hat{x}(t_i)$, and is not included in the equations related to filter tuning, i.e., Equations (4), (5), (6), and (8).

When comparing the error state estimates for the MMAE shown in Figure 65 to the error state estimates for the M³AE shown in Figure 66, it is clear that the M³AE does *not* significantly outperform the MMAE. In fact, the MMAE estimates are actually better for states 1 and 2. One of the conclusions stated by Miller [53] is that the M³AE will significantly outperform the MMAE *only* when there is significant *blending* of more than one elemental filter. Otherwise, the MMAE and M³AE might perform equally well, with no strong preference for the M³AE over the MMAE. This conclusion holds true for this case in which significant blending is not present (see Figure 64).

In contrast, Miller [53] anticipated that a moving-bank algorithm (which can be *centered* in the right neighborhood of the true parameter and *contracted* to a small enough span such that there *is* effective blending of more than one elemental filter) *would* provide the improvements possible with the M³AE architecture versus the MMAE, to a greater degree than a fixed-bank algorithm with a bank that spans the entire admissible parameter space for all time. Recall Figure 57 (case 1), which shows that the moving-bank decisions reduce the amount of the parameter space spanned by the filter-assumed parameter values. The result is *significant blending* between filters 3 and 4, as shown in Figure 58 and significant improvements in the latitude and longitude state estimates for the M³AE over the MMAE (see Figures 59 and 60).

One could further conjecture that blending two elemental filters in the scalar parameter case is more beneficial than blending three elemental filters in terms of M³AE versus MMAE performance, since the distance between \hat{a}_{MMAE} and the a_j 's for the two-filter case would be greater than the distance between \hat{a}_{MMAE} and the a_j 's for the three-filter case (particularly the middle of the three

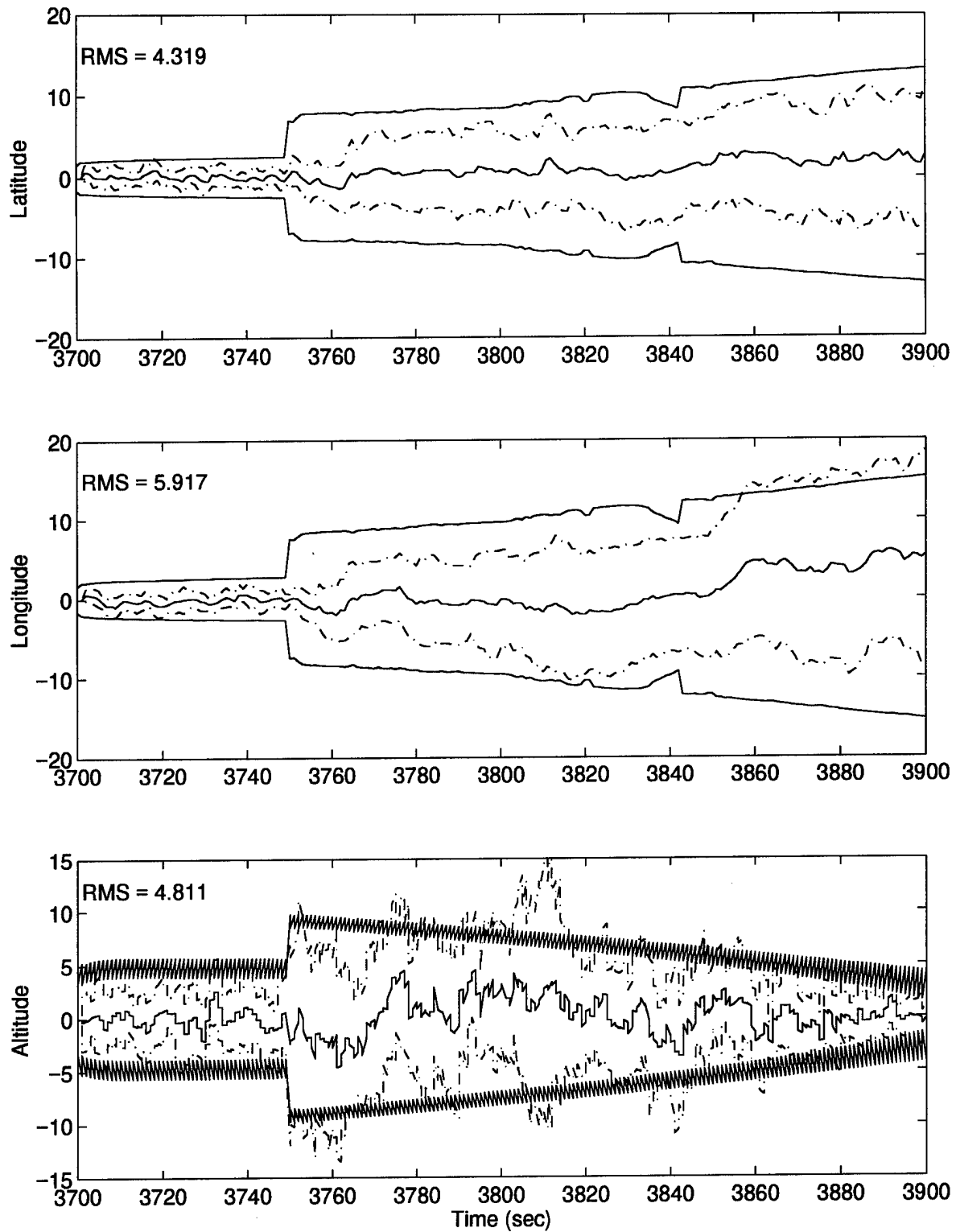


Figure 65. MMAE State Estimation Errors (feet) – Case 4: Density Algorithm

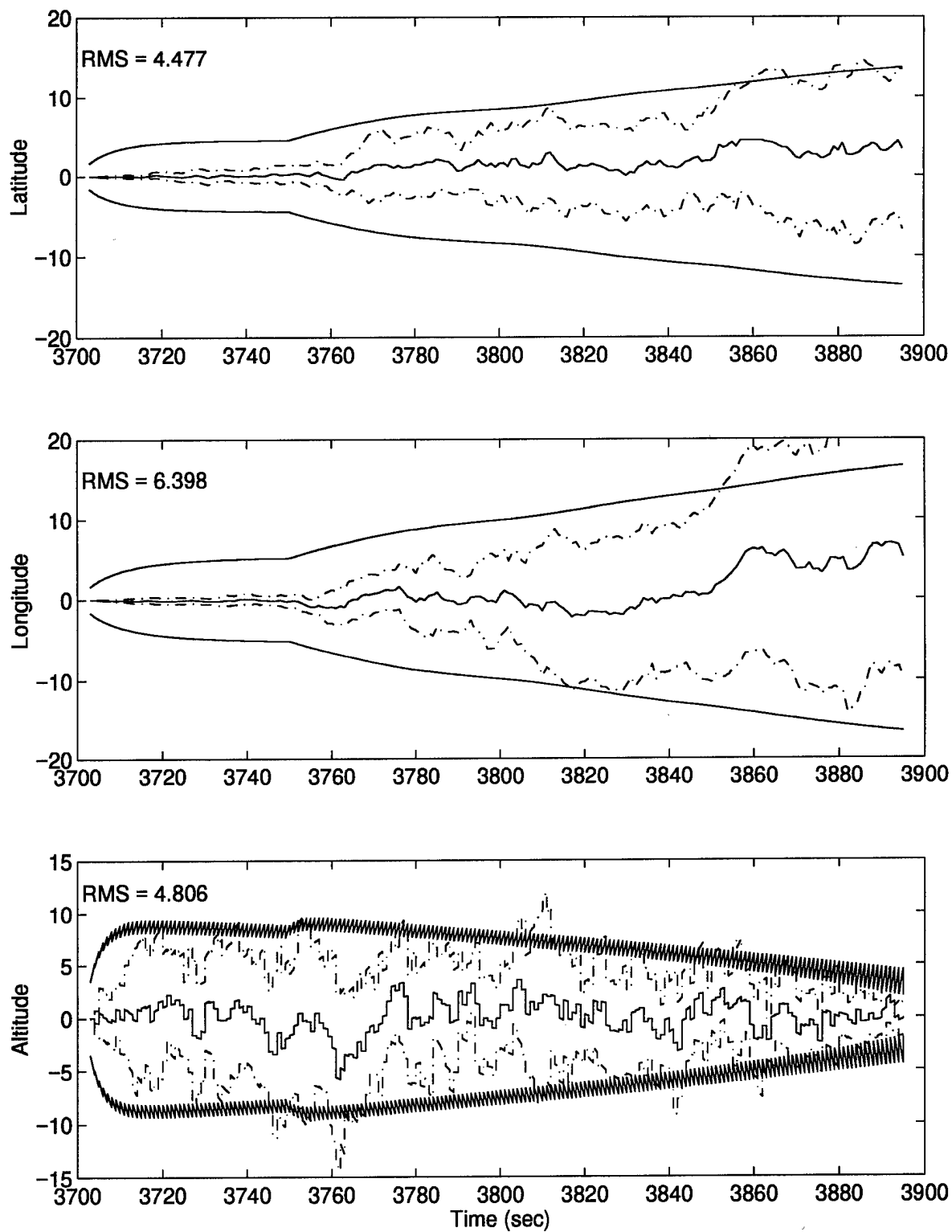


Figure 66. M³AE State Estimation Errors (feet) – Case 4: Density Algorithm

a_j 's). This conjecture requires validation through simulations and/or support through an analytical proof, which have not yet been accomplished.

Case 5: The density algorithm suffers erratic bank movements for this case, as seen by the parameter estimate plot in Figure 67. For example, the following series of movements results in oscillations of the parameter estimate about the true parameter value of $a_t = 1000$:

<u>Time (sec)</u>	<u>Movement Type</u>
3761	medium left
3764	soft right
3772	soft left
3784	soft right
3791	medium left
3793	soft right

The probability plot in Figure 68 indicates the MMAE's confusion about which filter is the best match to truth over the time frame $t = 3750$ sec to 3850 sec. The parameter estimation performance prior to and after this series of erratic movements is very good as the bank is contracted about the true parameter value. In order to account for the erratic behavior, the decision delay time was increased, as presented in the next section. The state estimation performance is shown for completeness in Figures 69 and 70.

4.7.3 Performance *With* Expansion and Additional Delay

Case 1: The parameter plot shown in Figure 71 illustrates the different decisions made by the density algorithm with expansions available and an increase in the decision delay from 1 to 5 sample periods. Specifically, the soft move right is delayed from $t = 3755$ sec with the basic density algorithm to $t = 3759$ sec due to the increased decision delay. Also, expansions occur at $t = 3781$ sec and 3831 sec, but the parameter estimate is seemingly unaffected by these decisions since filter 5 possesses most of the probability and its parameter value remains unchanged throughout the

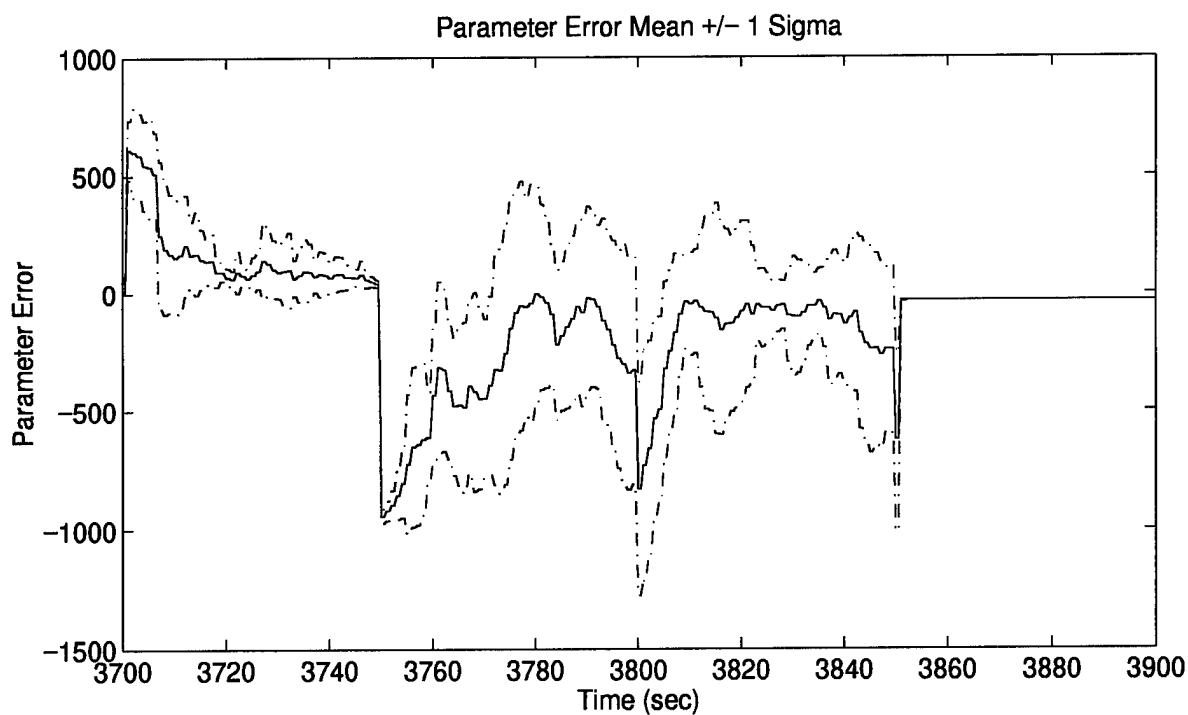
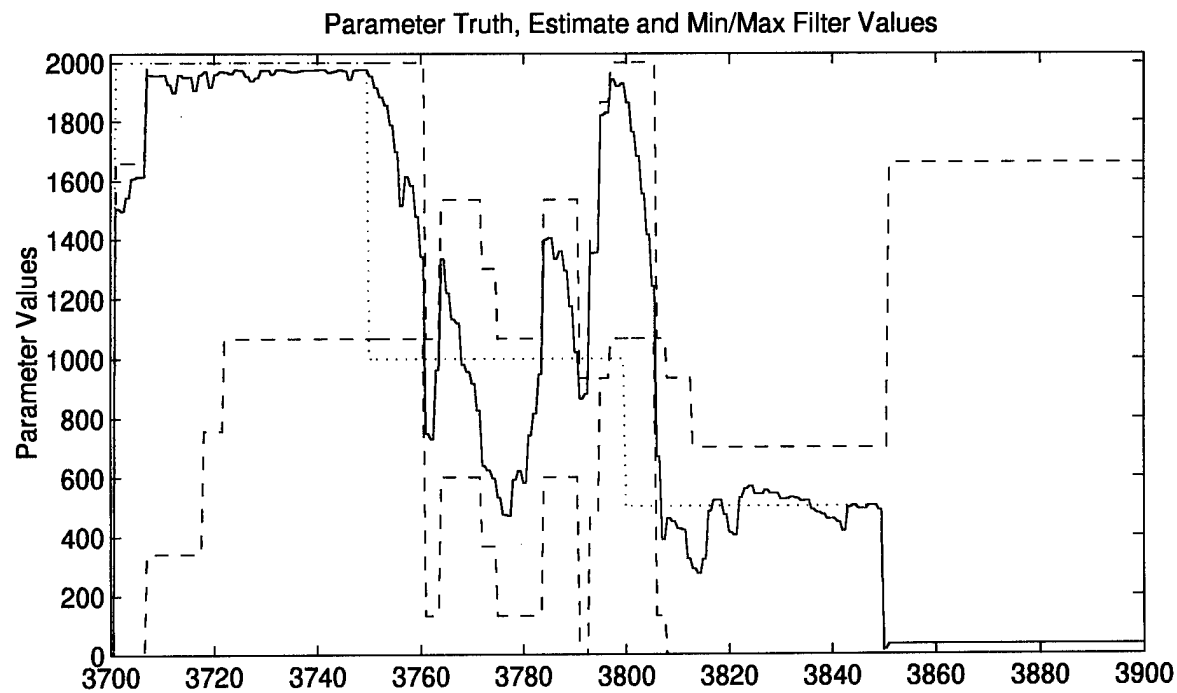


Figure 67. Parameter Estimation Performance – Case 5: Density Algorithm

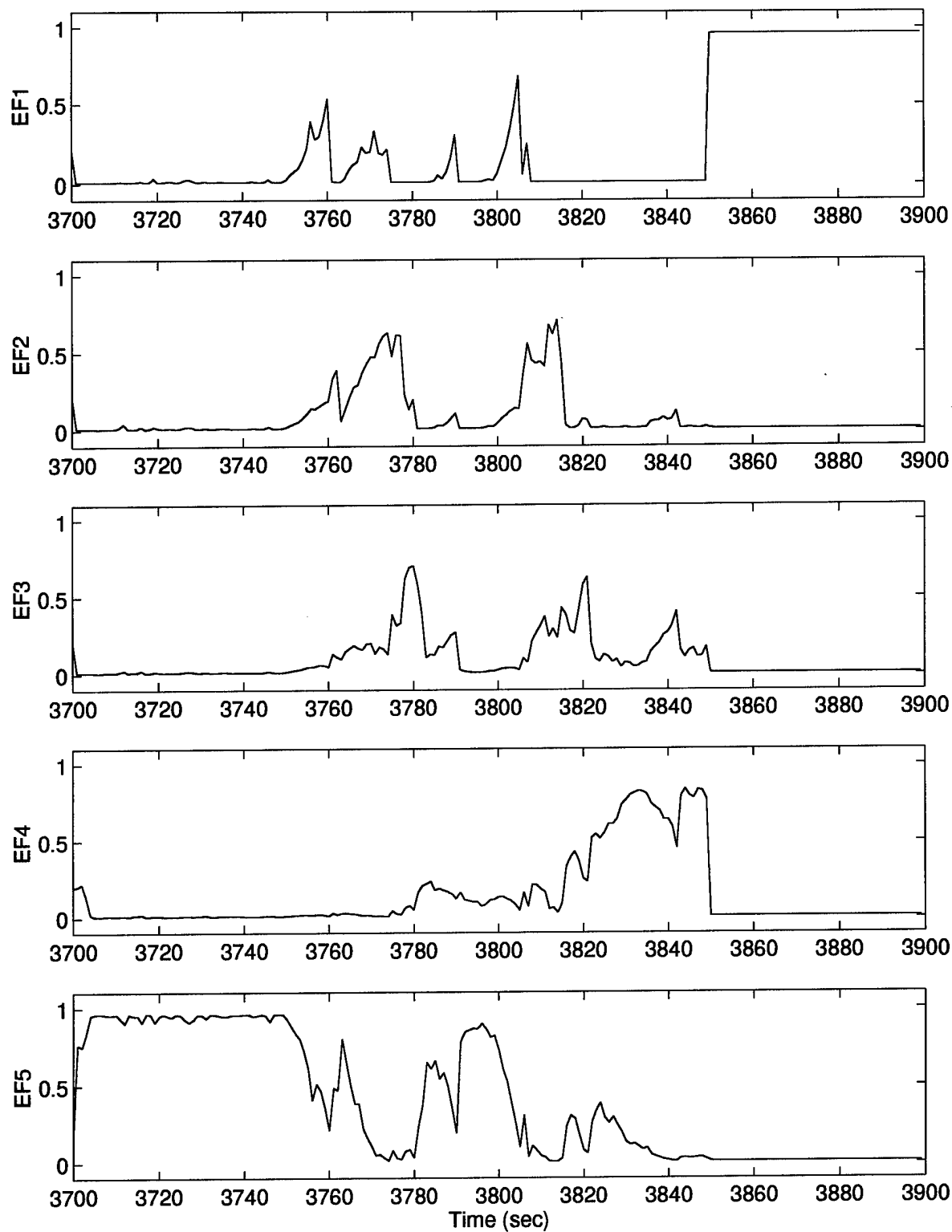


Figure 68. Elemental Filter Probabilities – Case 5: Density Algorithm

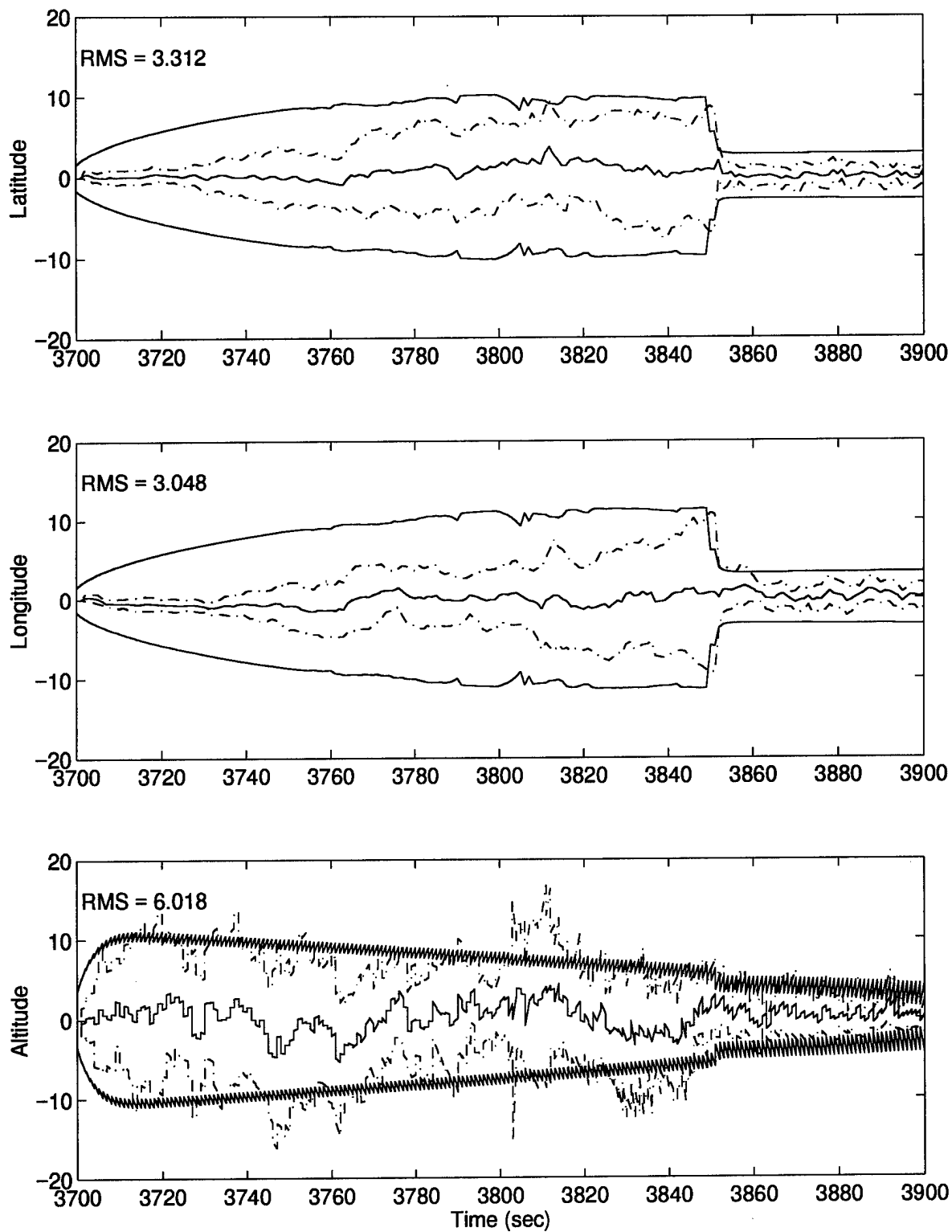


Figure 69. MMAE State Estimation Errors (feet) – Case 5: Density Algorithm

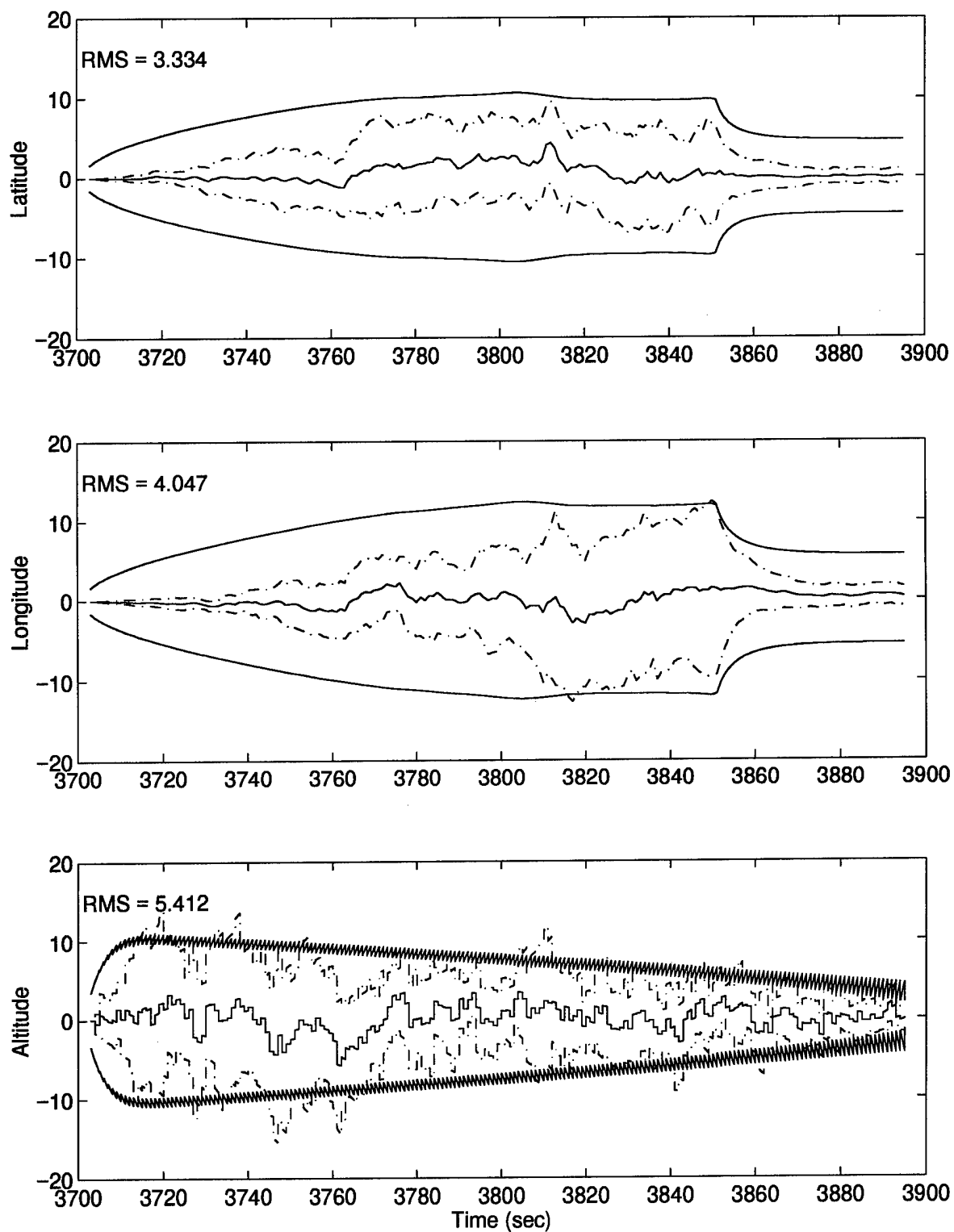


Figure 70. M^3 AE State Estimation Errors (feet) – Case 5: Density Algorithm

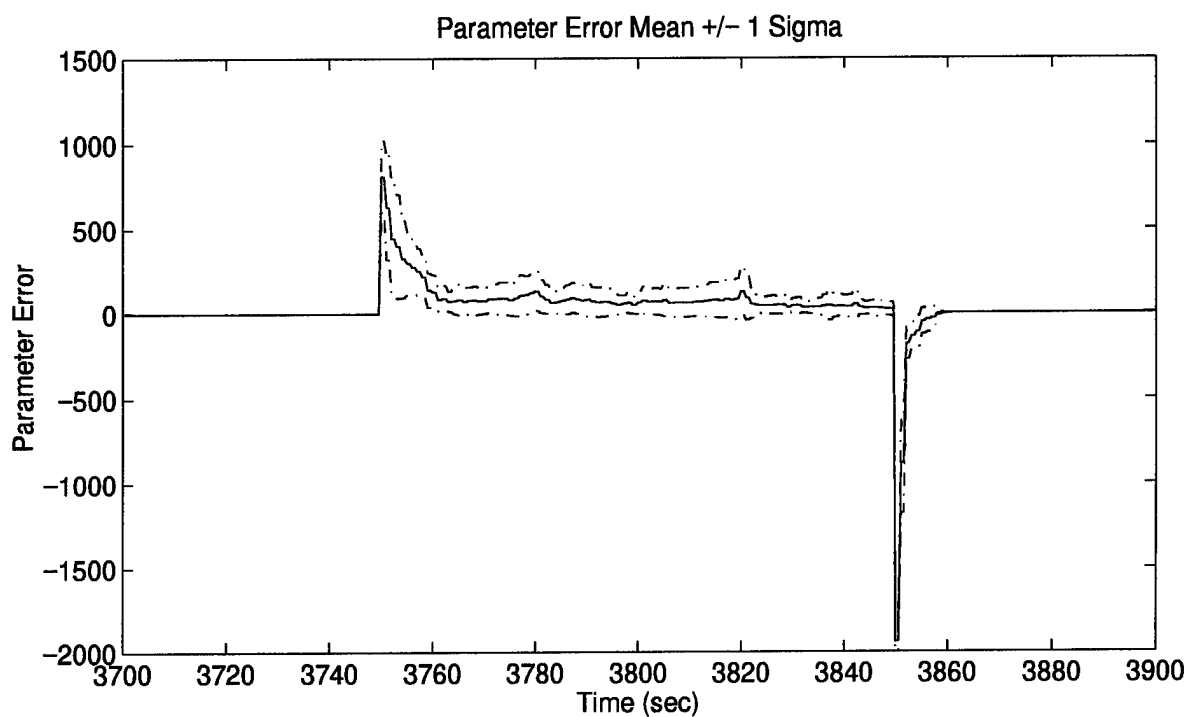
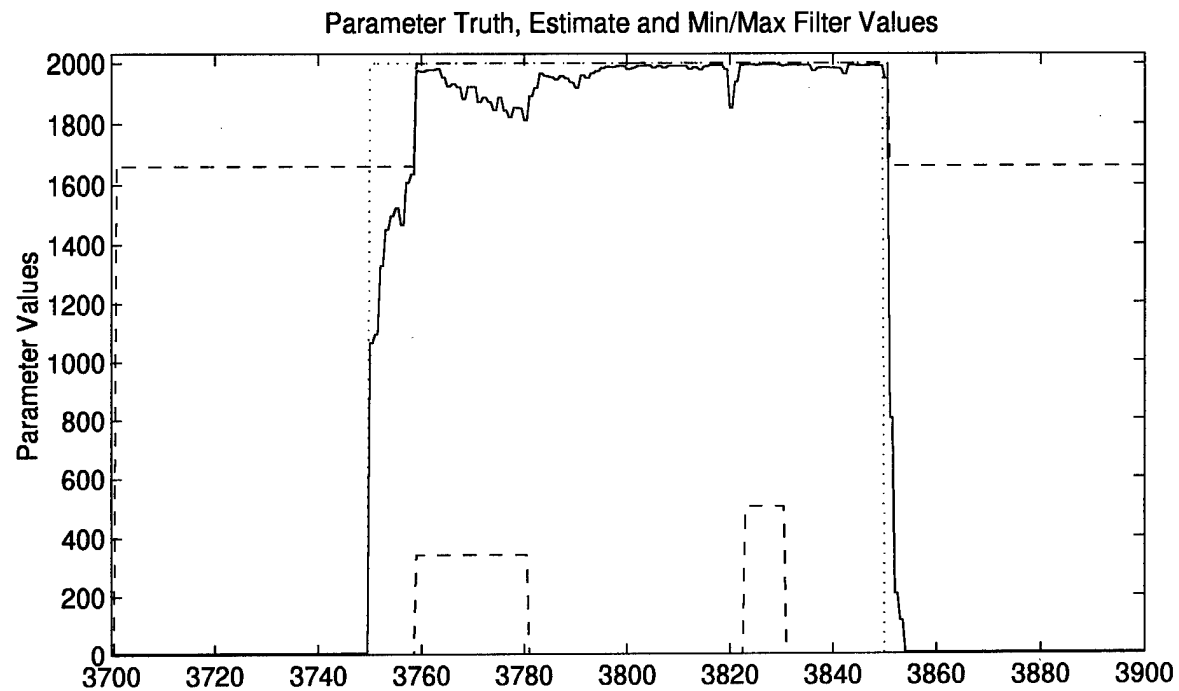


Figure 71. Parameter Estimation Performance – Case 1: Density Alg. with Expansions / Delay

expansions (see Figure 72). The trace pair (---) shows the expansion as a reduction in the minimum filter-assumed parameter value and no change in the maximum value.

Note the slight sluggishness induced in the parameter tracking as a result of the increased decision delay of 5 samples. This sluggishness is easily seen by comparing Figure 57 and 71 at the onset and removal of the interference/jamming, i.e., $t = 3750$ sec and 3850 sec. Also, note the slight increase in \hat{e}_{RMS}^a from 80 to 85, which is shown in Table 16 (Appendix E), for the density algorithm without and with the additional delay, respectively. Furthermore, the same transients in the MMAE error state estimates that plagued the fixed-bank MMAE for this case exist around $t = 3850$ sec. Again, this is due to the time needed by the MMAE to converge on the true parameter value. The final error state estimates provided by the M³AE approach are only slightly degraded as compared to the density algorithm performance without the additional delay (see Figure 73).

Cases 2 and 3: For these cases, the density algorithm performance is relatively unchanged by the addition of expansions and increased decision delays, and the results are not presented. The MMAE error state estimates still suffer some transient behavior once the interference/jamming is turned off, but the M³AE error state estimates are nearly identical.

Case 4: The increased decision delay prevents some of the erratic bank movement encountered with the basic density algorithm when a_t undergoes repeated small step increases, as discussed earlier. This is particularly evident when comparing the parameter estimate plots shown in Figures 63 and 74 at $t = 3818$ sec to 3845 sec for each version of the algorithm. However, the bank expansions induce similar erratic behavior as seen from the following sequence of decisions when the true parameter remains unchanged at $a_t = 2000$:

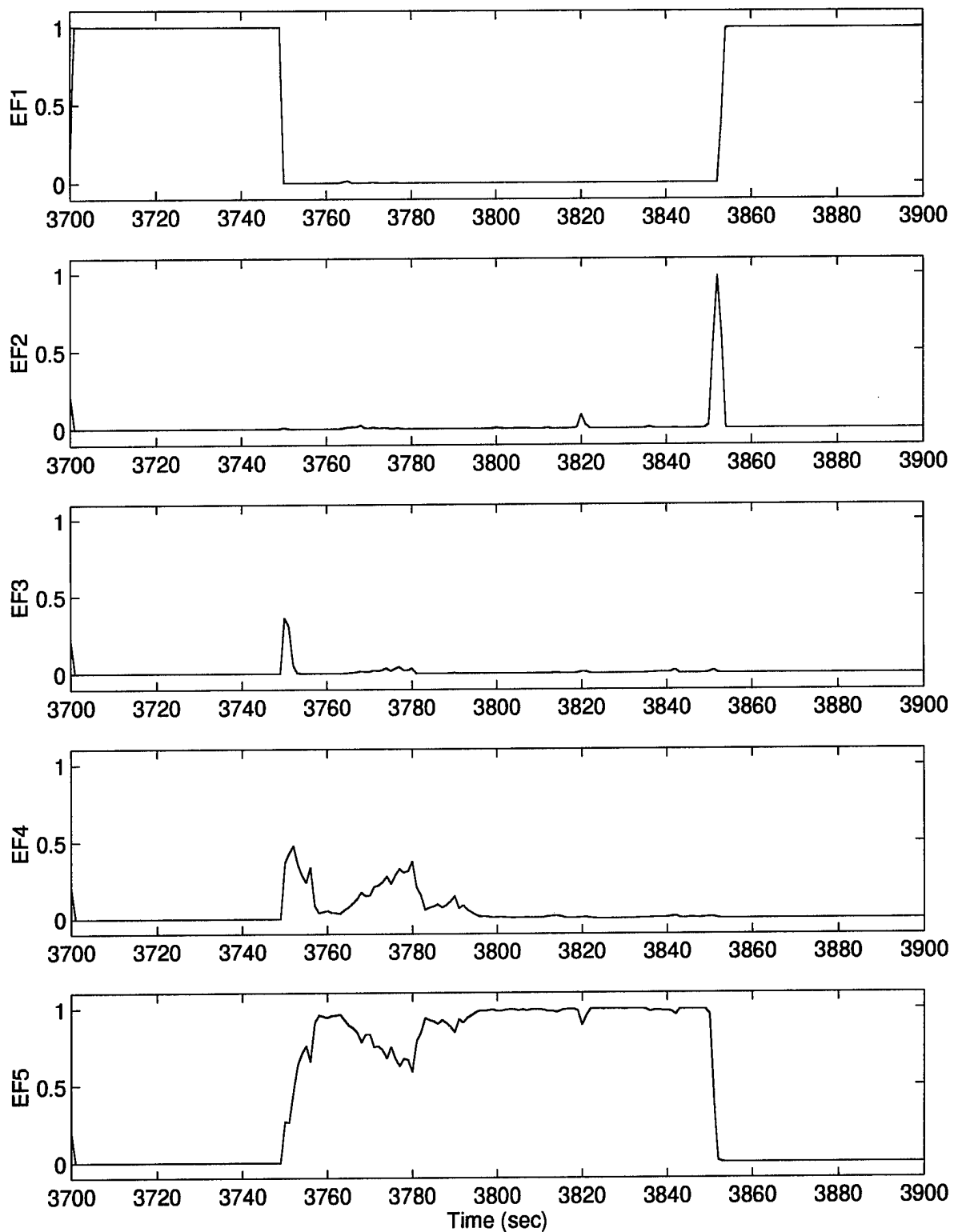


Figure 72. Elemental Filter Probabilities – Case 1: Density Alg. with Expansions / Delay

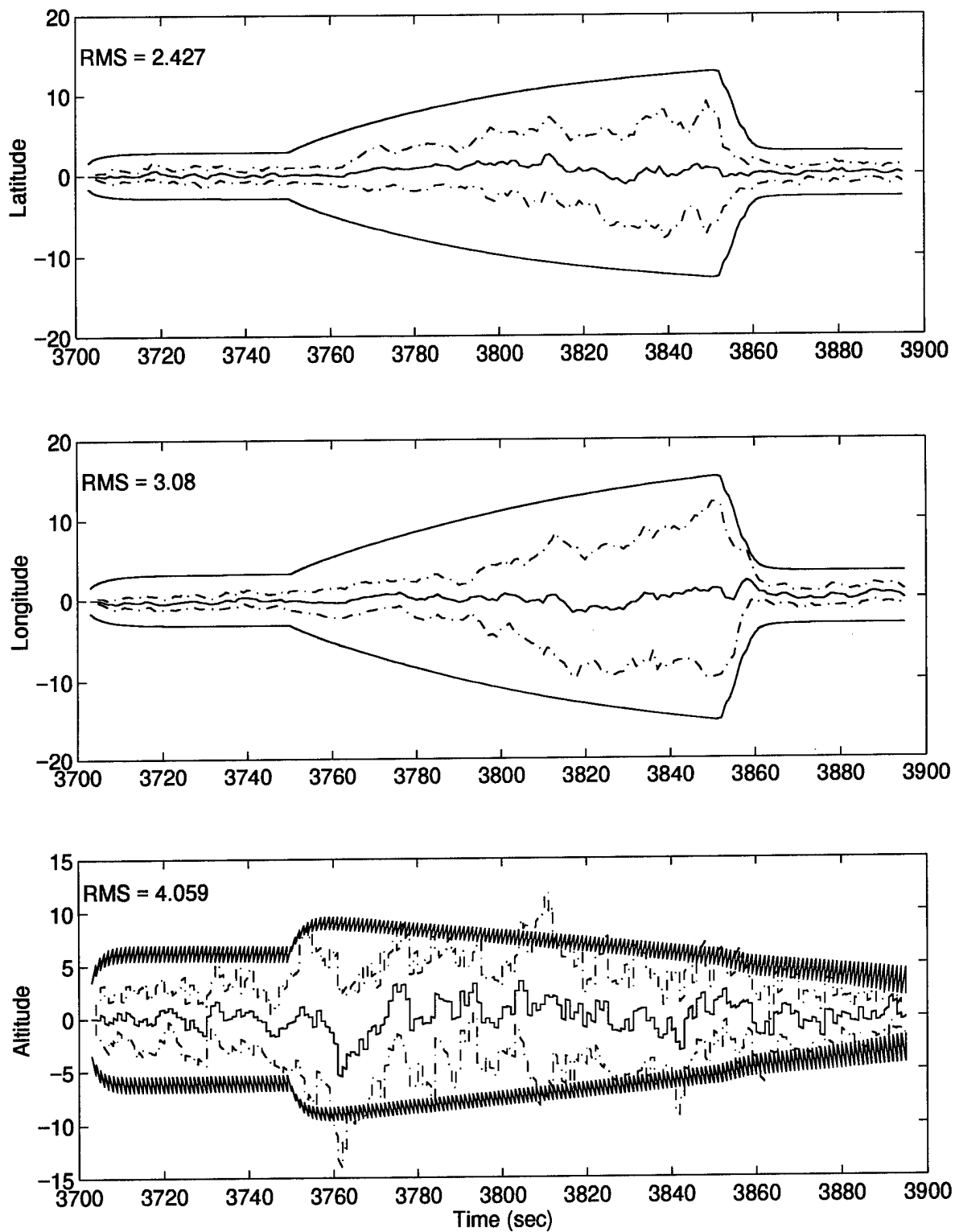


Figure 73. M³AE State Estimation Errors (feet) – Case 1: Density Alg. with Expansions / Delay

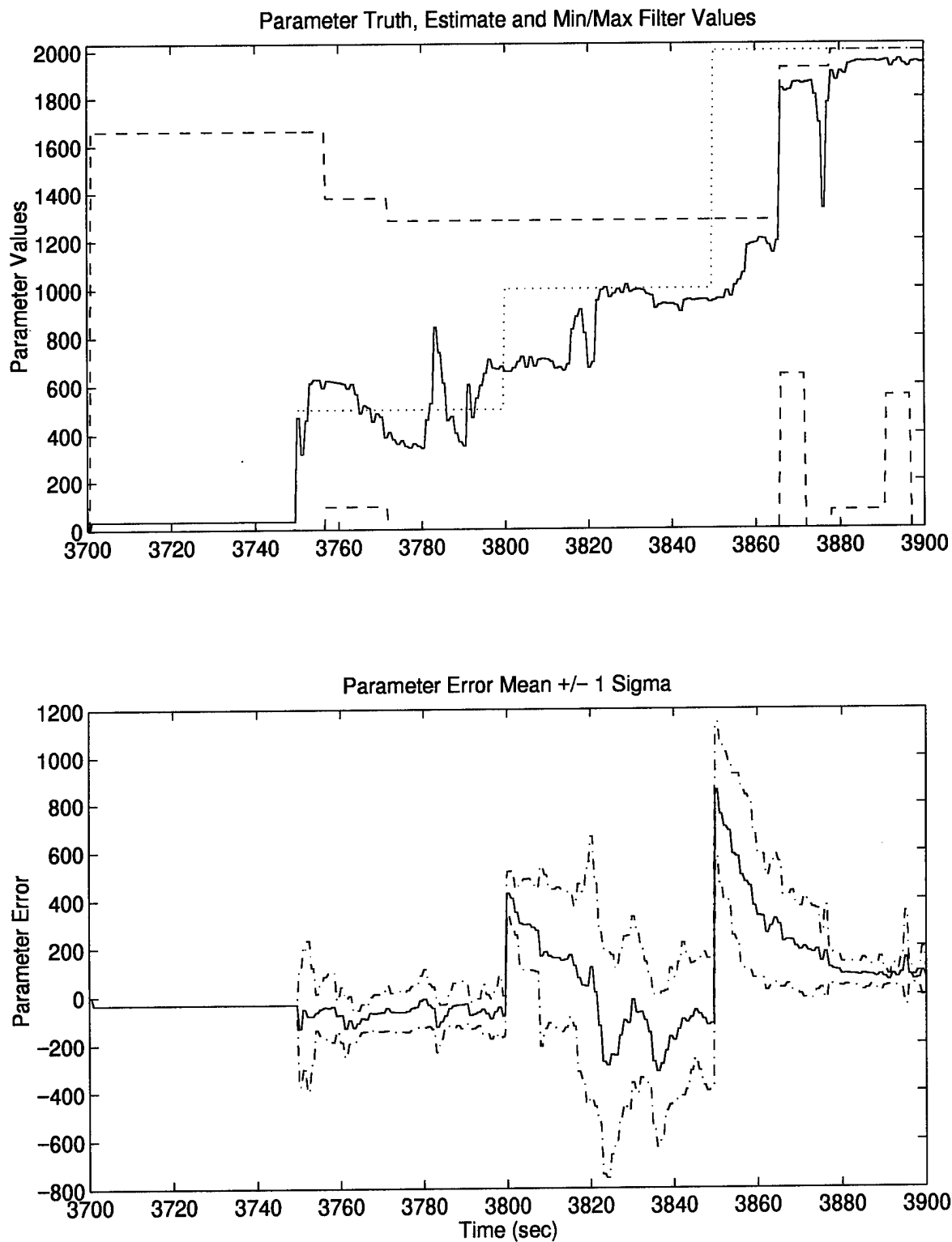


Figure 74. Parameter Estimation Performance – Case 4: Density Alg. with Expansions / Delay

<u>Time (sec)</u>	<u>Movement Type</u>
3866	soft move right
3872	expand
3878	soft move right
3891	contract
3897	expand

The state estimation performance is very similar for both versions of the density algorithm, as seen by comparing Figures 65 and 66 to Figures 75 and 76.

Case 5: Recall that the basic density algorithm suffered erratic bank movements when a_t undergoes a sequence of small step decreases, as described earlier for this case study. Comparison of Figures 67 and 77 shows the significant improvement gained by invoking the additional decision delay. This version of the density algorithm outperforms all other algorithms tested in terms of the parameter estimation measure \hat{e}_{RMS}^a .

4.8 MMAE Incorporating the Density Algorithm with Sheldon Discretization

4.8.1 Implementation Issues

Section 3.1.5.4 detailed the concept of combining the decision making process of the density algorithm with the discretization process of the on-line Sheldon algorithm. As before, five elemental filters are used with a lower bound on the probabilities of $p_{\min} = 0.001$. A look-up table of parameter values is created off-line using the modified Sheldon algorithm for optimal parameter estimation with finite horizon. The density algorithm provides the Sheldon algorithm with two endpoints over which to discretize the five filter-assumed parameter values. Each pair of endpoints and their associated five filter-assumed parameter values constitute a seven-element row in the table, i.e., there are seven columns in the table. Recall that the admissible parameter range is $1 \leq a_j \leq 2000$, so each entry (row) in the table must assume values for the endpoints within this range. In order to account for every possible endpoint value, a table with an infinite number of entries would be needed.

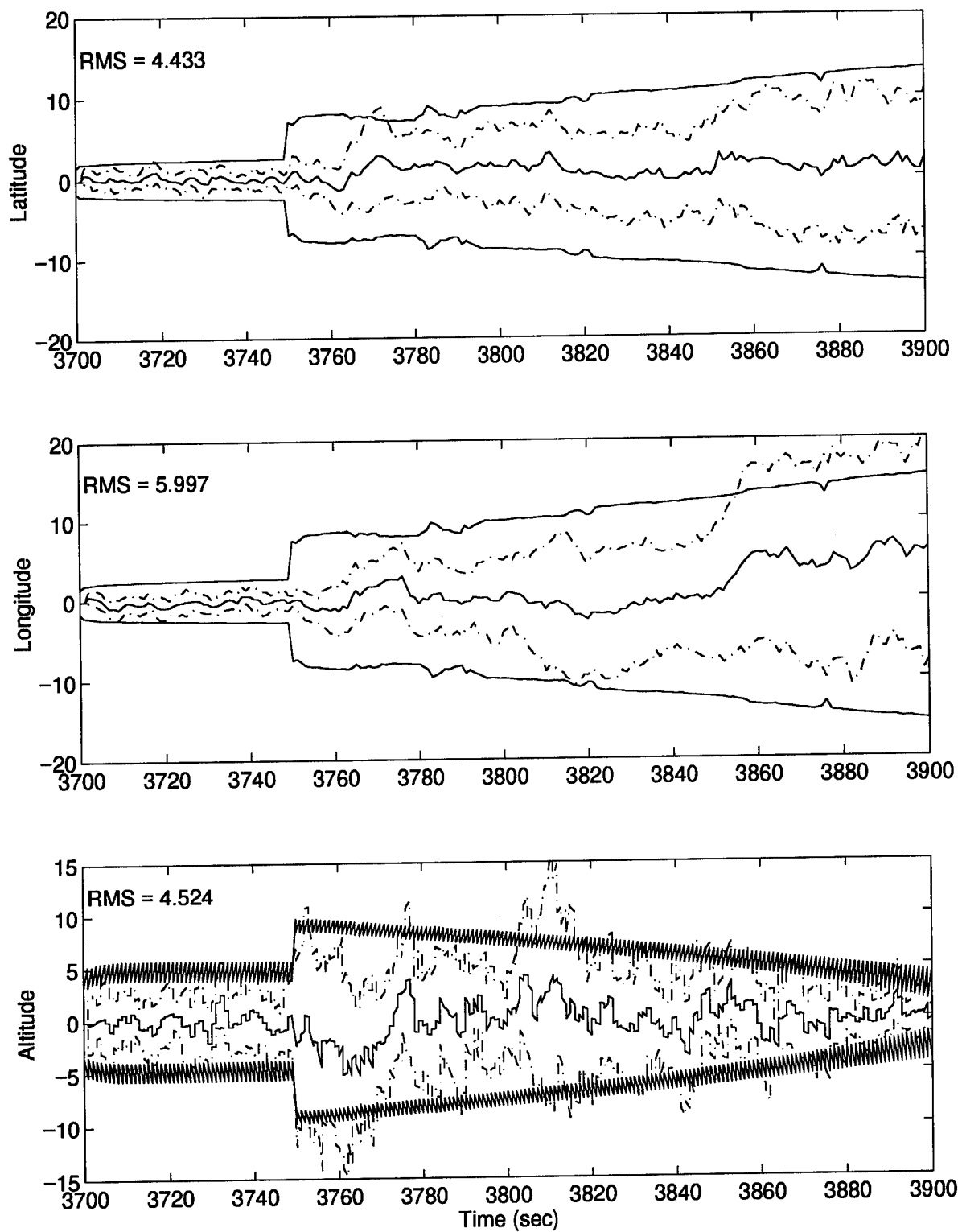


Figure 75. MMAE State Estimation Errors (feet) – Case 4: Density Alg. with Expansions / Delay

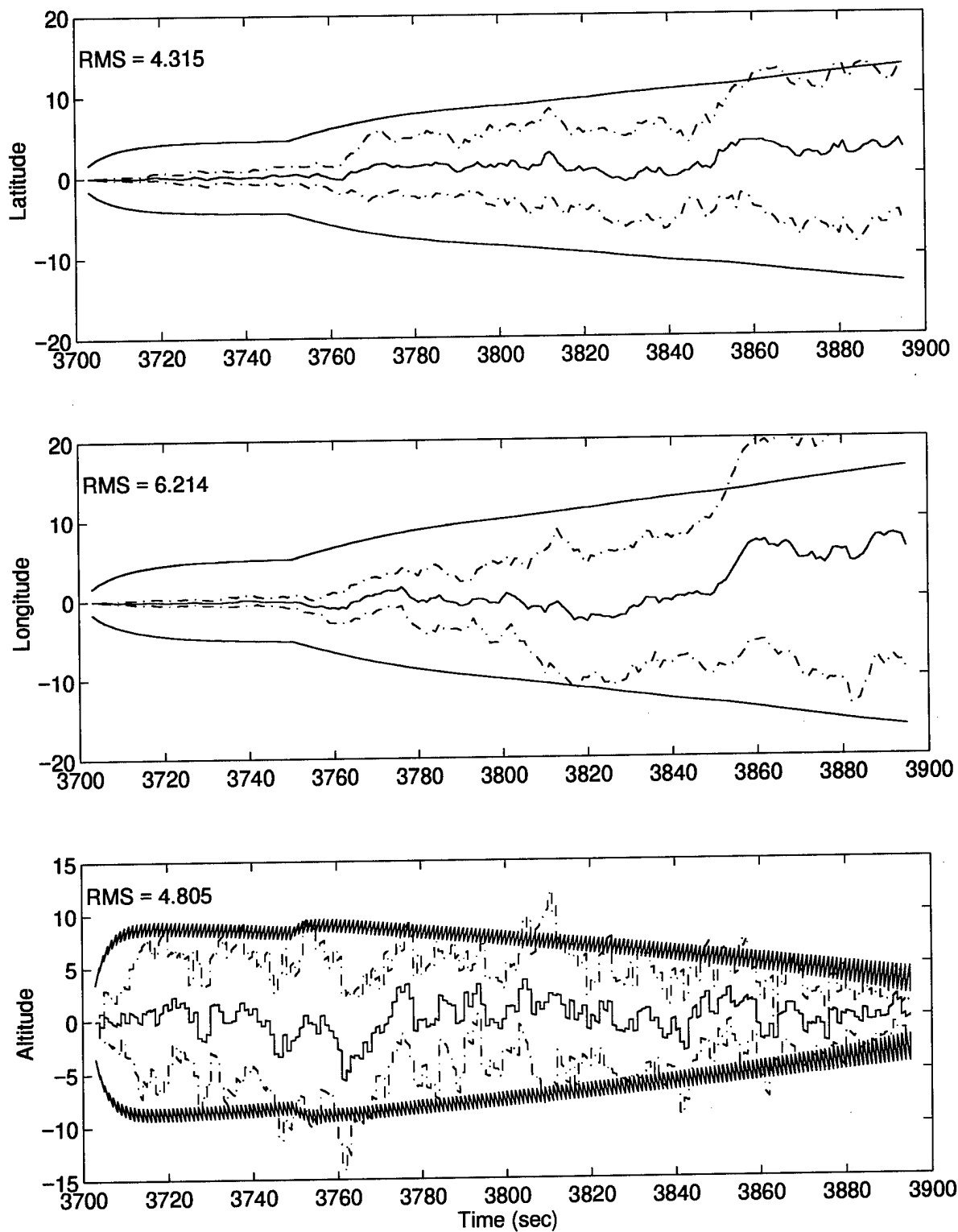


Figure 76. M³AE State Estimation Errors (feet) – Case 4: Density Alg. with Expansions / Delay

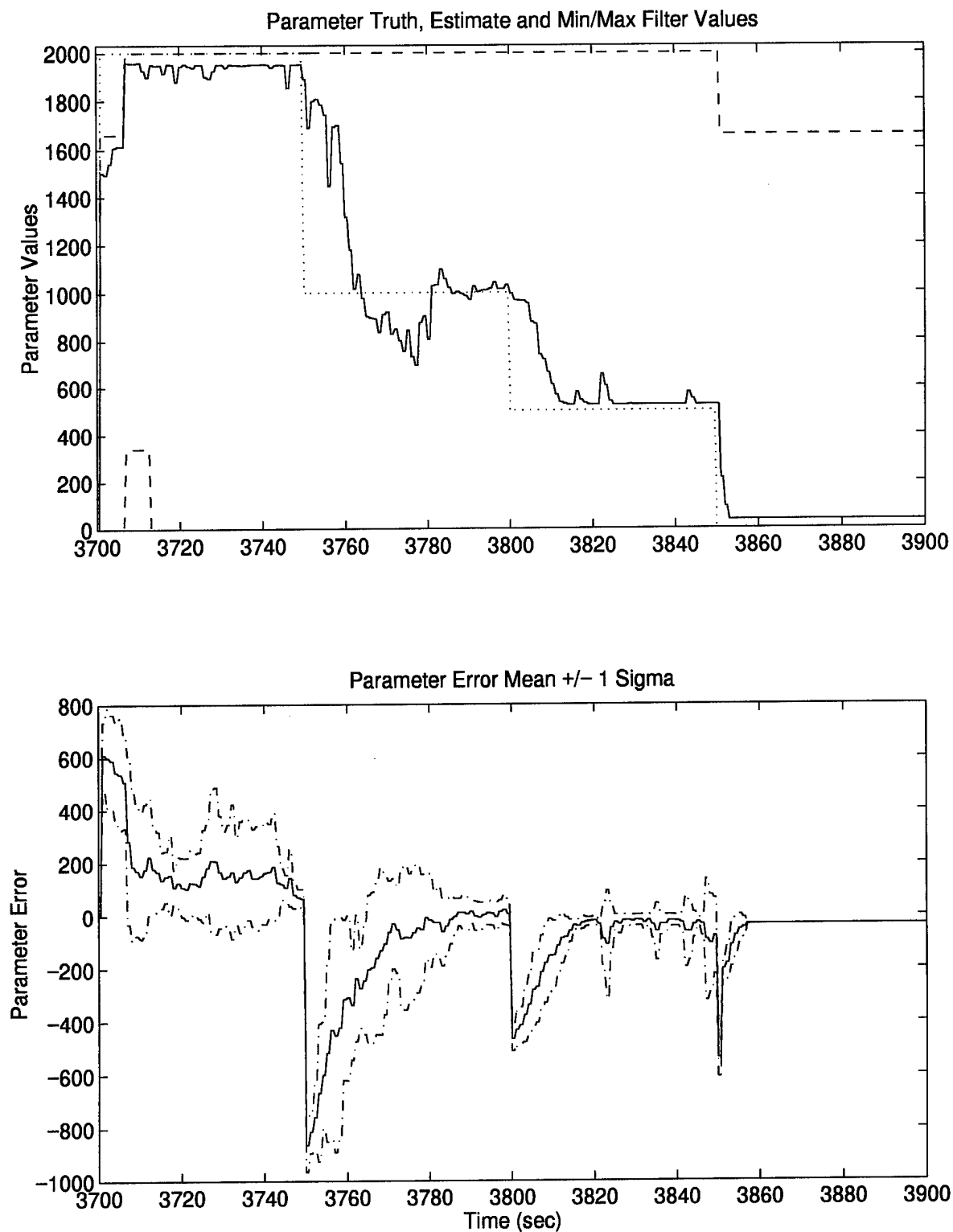


Figure 77. Parameter Estimation Performance – Case 5: Density Alg. with Expansions / Delay

Since this is not practical, discrete values for the endpoints are arbitrarily chosen, beginning with uniformly spacing the endpoints over the parameter range for simplicity. First, 18 left endpoints are chosen over the range $9ft^2 \leq R_{GPS} \leq 17,000ft^2$ in increments of $1000ft^2$. This depicts $R_{GPS}^1 = a_1 R_0$ over the parameter range $1 \leq a_1 \leq 1889$ in increments of 111 where the superscript on R_{GPS}^1 indicates the filter number. The increment value of $1000ft^2$ is purely *ad hoc* with the goal of selecting a reasonable number of table entries that are not too far apart. A smaller increment size might be necessary for problems with higher sensitivity to the parameter value. Note the largest value for the left endpoint is arbitrarily chosen less than a_{\max} since the density algorithm will not select a left endpoint equal to a_{\max} . For each left endpoint, right endpoints are selected over the range $R_{GPS}^1 + 2000ft^2 \leq R_{GPS}^5 \leq 18,000ft^2$ in increments of $1000ft^2$ (again chosen in an *ad hoc* fashion and depicting $R_{GPS}^5 = a_5 R_0$). The two exceptions are (1) with $R_{GPS}^1 = 9ft^2$, the first right endpoint is $2000ft^2$ versus $2009ft^2$ and (2) with $R_{GPS}^1 = 17,000ft^2$, the only right endpoint is $18,000ft^2$. For example, with a left endpoint of $R_{GPS}^1 = 9ft^2$ there are 17 associated right endpoints (and thus 17 table entries or rows) with values of $R_{GPS}^5 = 2000, 3000, \dots, 18000ft^2$. Similarly, with $R_{GPS}^1 = 15,000ft^2$ there are two associated right endpoints (and thus 2 table entries or rows) with values of $R_{GPS}^5 = 17,000ft^2$ and $18,000ft^2$. Additionally, the smallest value for the right endpoints is arbitrarily chosen greater than a_{\min} since the density algorithm will not select a right endpoint equal to a_{\min} . The total number of right endpoints is $17 + 16 + \dots + 2 + 1 + 1 = 154$, resulting in 154 entries or rows in the table. Therefore, the table size is 154-by-7 parameter values.

The design parameters used here are the same as those used for the basic density algorithm summarized in Table 6 (page 178). This allows for a direct comparison of the pros and cons of implementing the Sheldon discretization over the simple uniform spacing techniques used in the basic density algorithm. The performance section which follows (see Section 4.8.2) indicates the inability of this algorithm to react to gradual decreases in the level of interference/jamming. This

motivated the use of expansions (see Section 4.8.3) which resolve the problem but require the same longer decision delay time of 5 sample periods used with the basic density algorithm in Section 4.7.3 to counter the erratic bank changes.

4.8.2 Performance *Without* Expansion and Additional Delay

Case 1 : Figure 78 shows the performance of this algorithm in terms of the parameter estimate. The traces for the minimum and maximum filter-assumed parameter values (---) show a series of five soft moves to the right at times $t = 3755, 3764, 3782, 3784$, and 3792 sec, respectively, as determined by the density algorithm. The resulting parameter estimate is steadily increased but consistently lower than truth because it is upper bounded by the parameter value assumed by filter 5. This upper bound comes directly from the Sheldon discretization which never chooses a parameter value for a filter equal to the maximum admissible parameter value. The cost minimization process precludes such a choice since doing so would result in larger average values for the mean squared estimation error given by Equation (60). This is discussed further in cases 4 and 5, and a simple solution is presented that allows the bank to traverse the *entire* range of parameter values. The same trends in the MMAE and M^3 AE error state estimates identified for the density algorithm *without* expansions are present here, so the nearly identical plots are not shown. The \hat{e}_{RMS}^x values also indicate the similarity in their performance.

Case 2 : The decisions made here by the density algorithm *with* the Sheldon discretization (a single contraction at $t = 3757$ sec) parallel those for the density algorithm *without* the Sheldon discretization. However, the parameter estimate is consistently biased low, as seen in Figure 79, or biased high for other sample runs due to the choices of the filter-assumed parameter values provided by the Sheldon algorithm. Although the MMAE *tends* to favor the more conservatively tuned filters, as discussed in Section 3.1.5.1, it will assign greater probability to less conservatively tuned filters

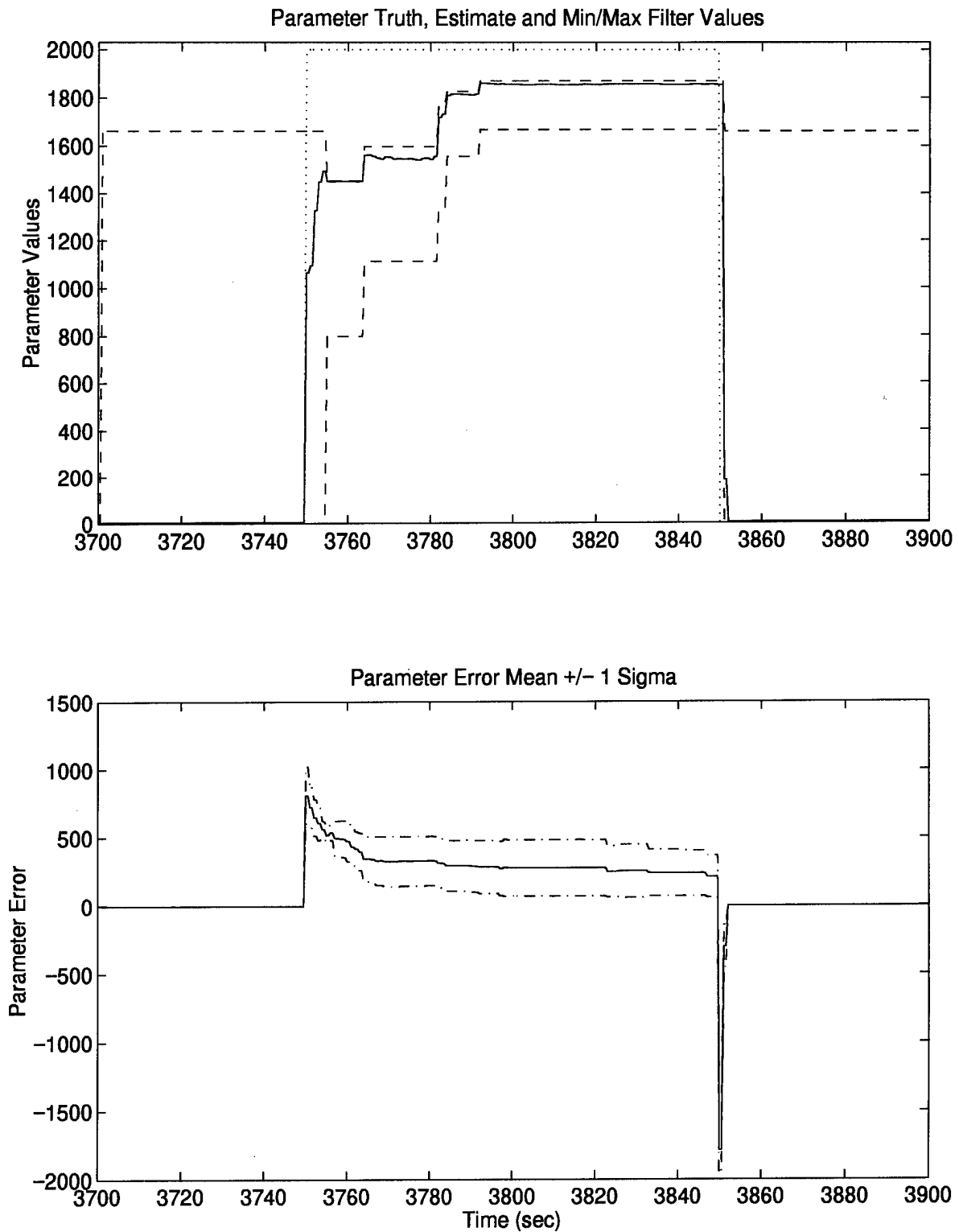


Figure 78. Parameter Estimation Performance – Case 1: Density / Sheldon Alg.

on occasion, resulting in a biased low parameter estimate for a given sample run. The parameter error mean in Figure 79 is primarily negative when the interference is present, illustrating that a bias high was more common over the 10 Monte Carlo runs. In fact, observation of the 10 runs reveals that six of the 10 runs followed the tendency to bias the parameter estimate high.

As with the fixed-bank algorithm, the Sheldon discretization often chooses parameter values for the filters which lie above or below a_t rather than very close to it. This results in a bias in the parameter estimate not often encountered with the basic density algorithm which uses uniform discretization. Recall that soft moves, expansions and contractions dictated by the basic density algorithm all center around the filter having the maximum probability. This provides the opportunity for the center filter to become well matched to truth after a series of such decisions. The Sheldon discretization essentially ignores the maximum probability filter since a cost minimization is performed over a parameter range (a_{\min} to a_{\max}), which is dictated by the density algorithm, to determine the new filter-assumed parameter values. This will not, in general, lead to a filter-assumed value that matches truth, as discussed more fully in case 3.

Case 3 : Unlike the results in case 2 above, the Sheldon discretization combined with the density algorithm outperforms the basic density algorithm with uniform discretization for this case. The parameter estimate and probability plots shown in Figures 80 and 81 indicate that the Sheldon discretization made a fortuitous choice for the parameter value assumed by filter 3 as it matches a_t very well when interference/jamming is present. This is largely coincidental but motivates a recommendation to take the parameter values chosen by a Sheldon discretization and set the one closest to \hat{a}_{MMAE} equal to \hat{a}_{MMAE} . This has the potential of overcoming the bias problem encountered with case 2 above while taking advantage of the cost minimization provided by Sheldon's algorithm. This will be discussed further in Chapter 5.

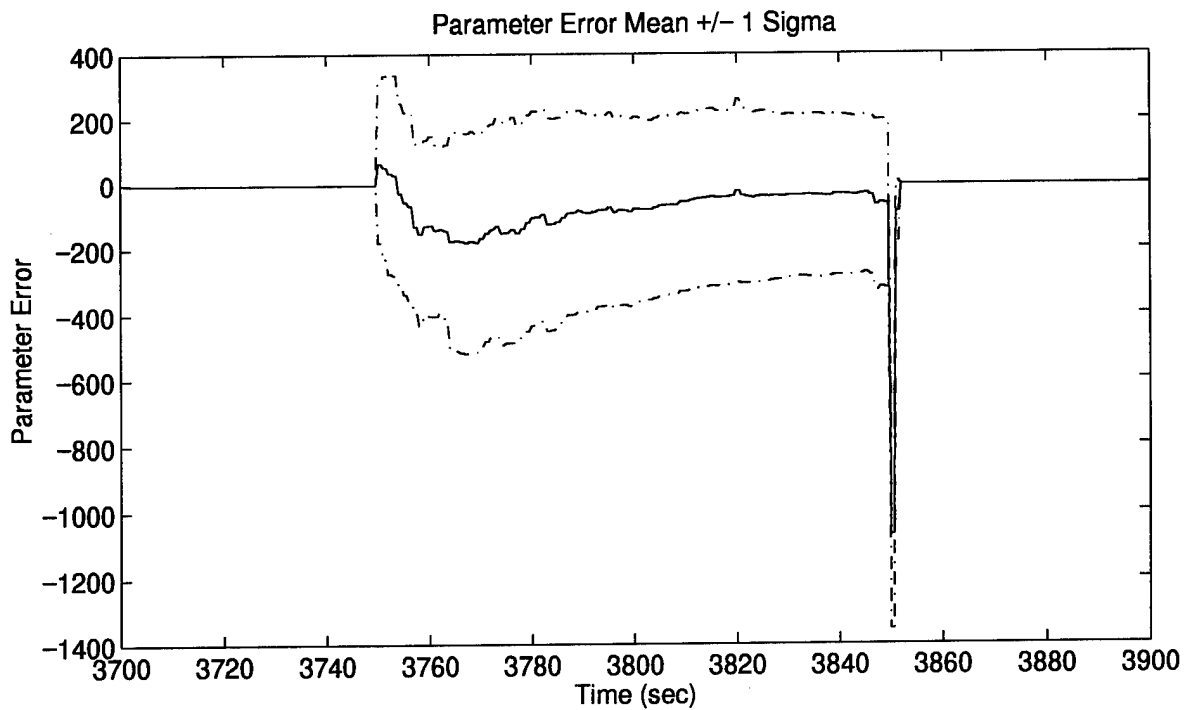
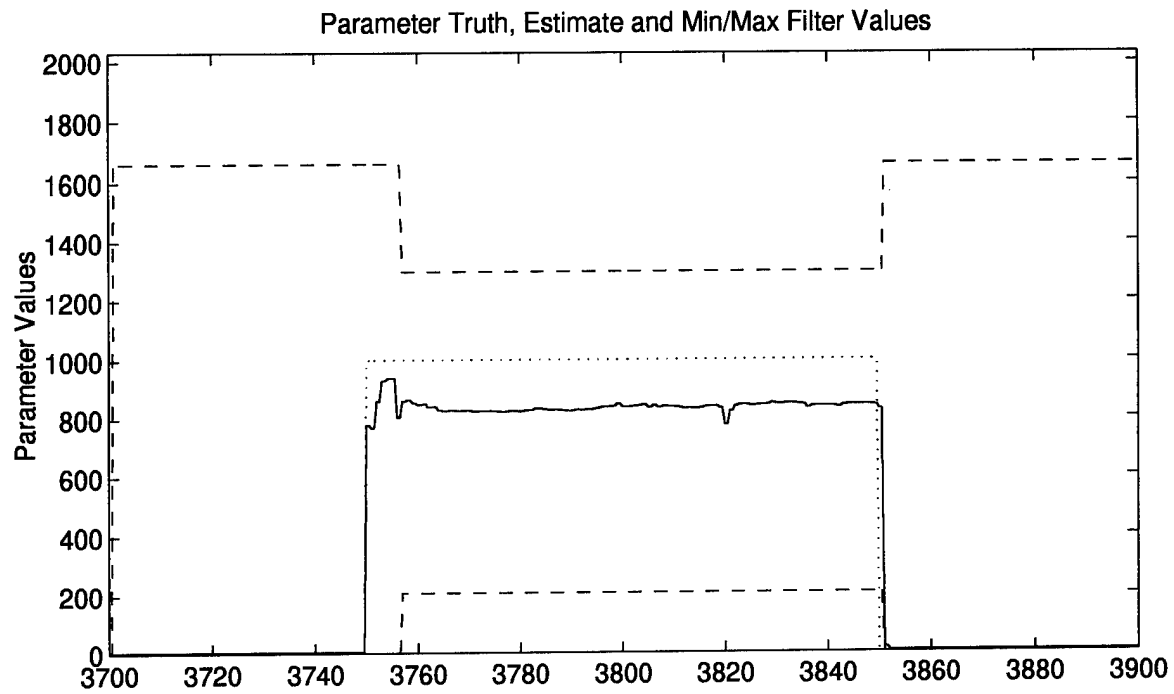


Figure 79. Parameter Estimation Performance – Case 2: Density / Sheldon Alg.

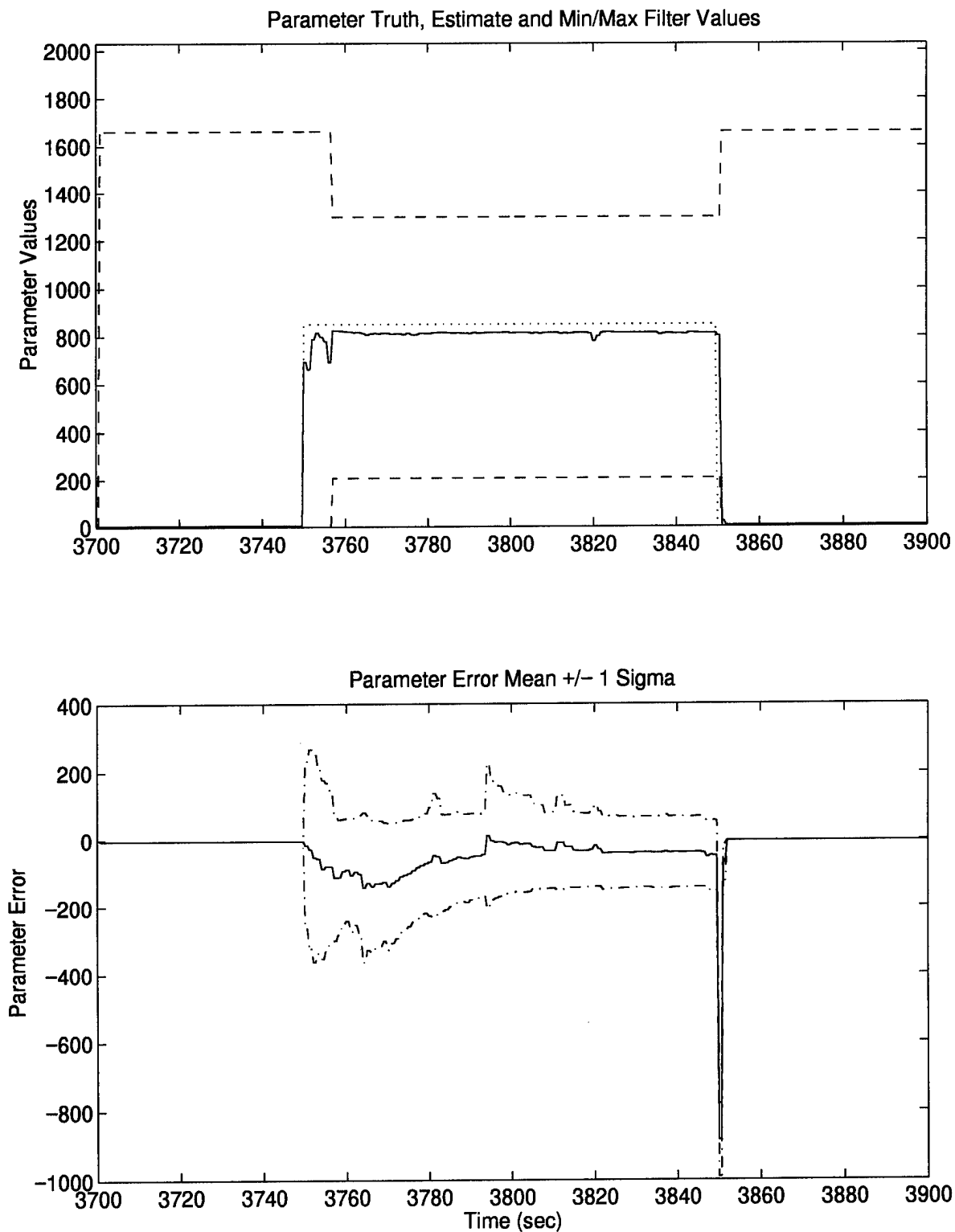


Figure 80. Parameter Estimation Performance – Case 3: Density / Sheldon Alg.

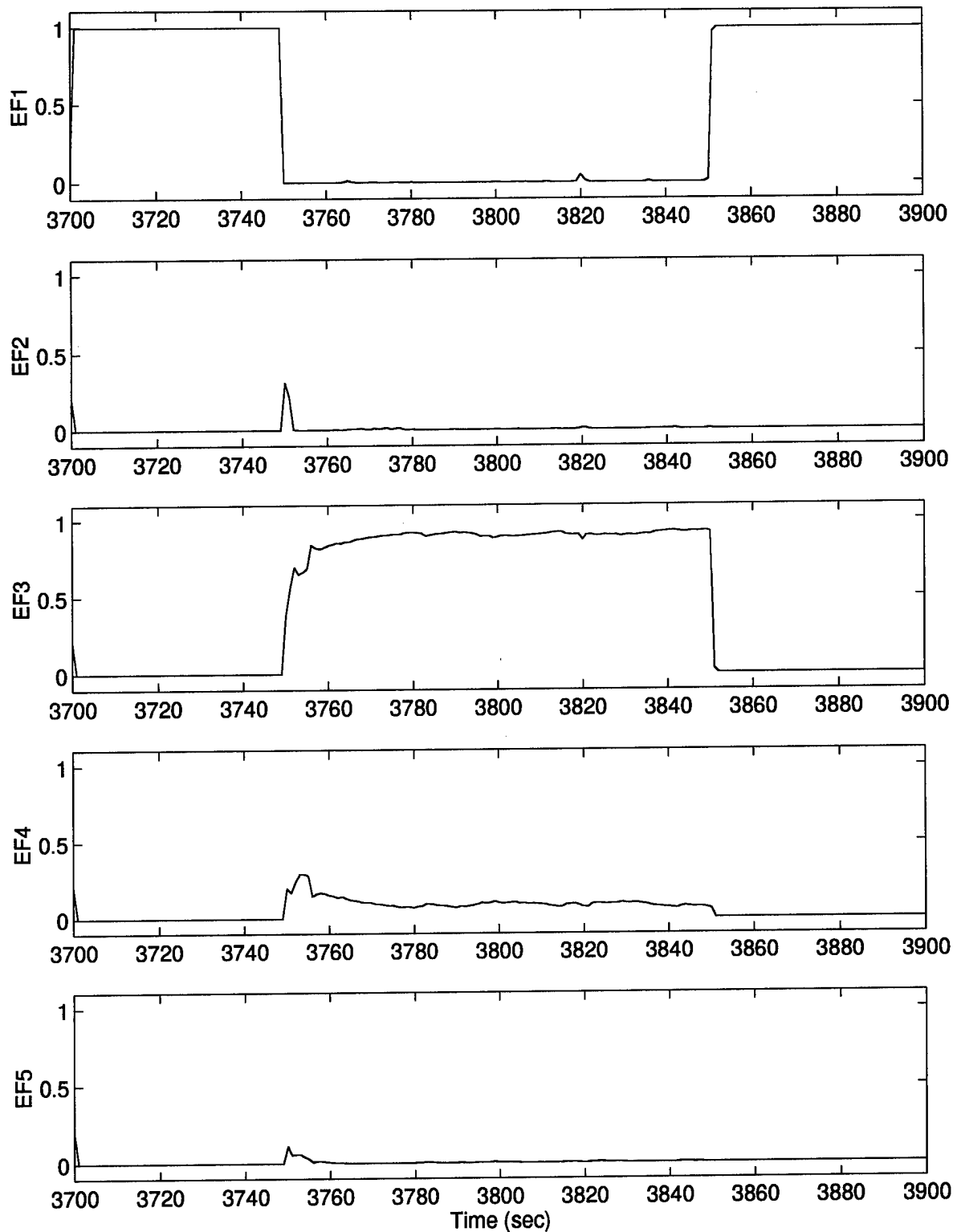


Figure 81. Elemental Filter Probabilities – Case 3: Density / Sheldon Alg.

Case 4: This case identifies an interesting attribute of the combined density algorithm with Sheldon discretization which is referred to as “movement-induced contractions.” The density algorithm determines when a bank movement is to be made and passes the endpoints of the new bank to the Sheldon discretization process, which in turn provides J new filter-assumed parameter values. Since the Sheldon discretization process does not in general choose these endpoints as two of the new parameter values, the new filter bank will actually cover a smaller portion of the parameter space, resulting in a bank contraction. For example, given the endpoints $a_{\min} = 1000$ and $a_{\max} = 6000$, the Sheldon chosen parameter values for filters 1 – 5 are $[a_j; j = 1, \dots, 5] = [1276 \ 2167 \ 3698 \ 5181 \ 5783]$. An alternative approach would be to constrain the Sheldon optimization routine such that the endpoints dictate two of the chosen parameter values (i.e., fix two of the parameter values chosen by the Sheldon routine as these endpoints) and only the inner three ($J - 2$) parameter values can affect the cost minimization. This could be viewed as defeating the purpose of allowing the Sheldon routine to optimize the parameter choices and was not implemented for these studies. However, future researchers may want to consider this idea, as recommended in Chapter 5.

Evidence of the movement-induced contractions is shown in Figure 82 by observing the minimum and maximum filter-assumed parameter values indicated by the pair of traces (---). A series of four soft moves right occurs between $t = 3805$ sec and 3817 sec, resulting in significant contraction of the filter bank and a maximum filter-assumed parameter value, a_5 , which is below a_t . This trend continues throughout the simulation, causing the parameter estimate to be consistently biased low.

Case 5: The movement-induced contractions described under case 4 causes severe problems in estimating the parameter for case 5. Notice a series of four moves right from $t = 3707$ sec to 3718 sec in Figure 83. Although movement to the right is desirable in this case, the induced contractions cause two problems.

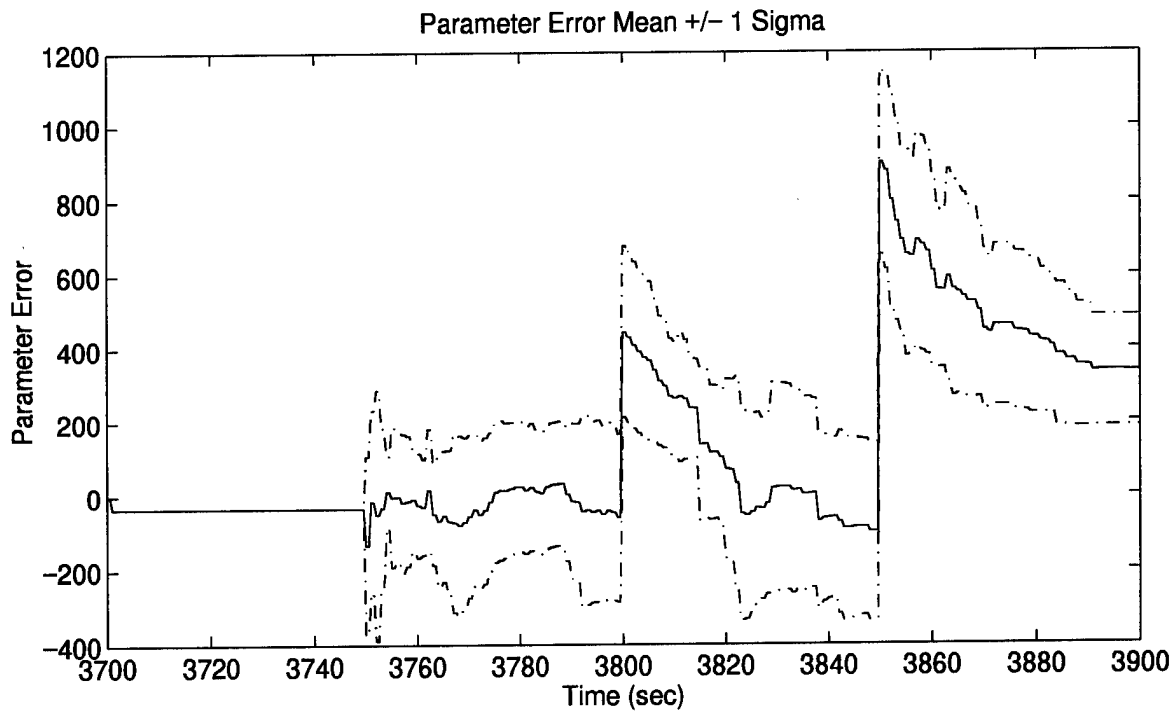
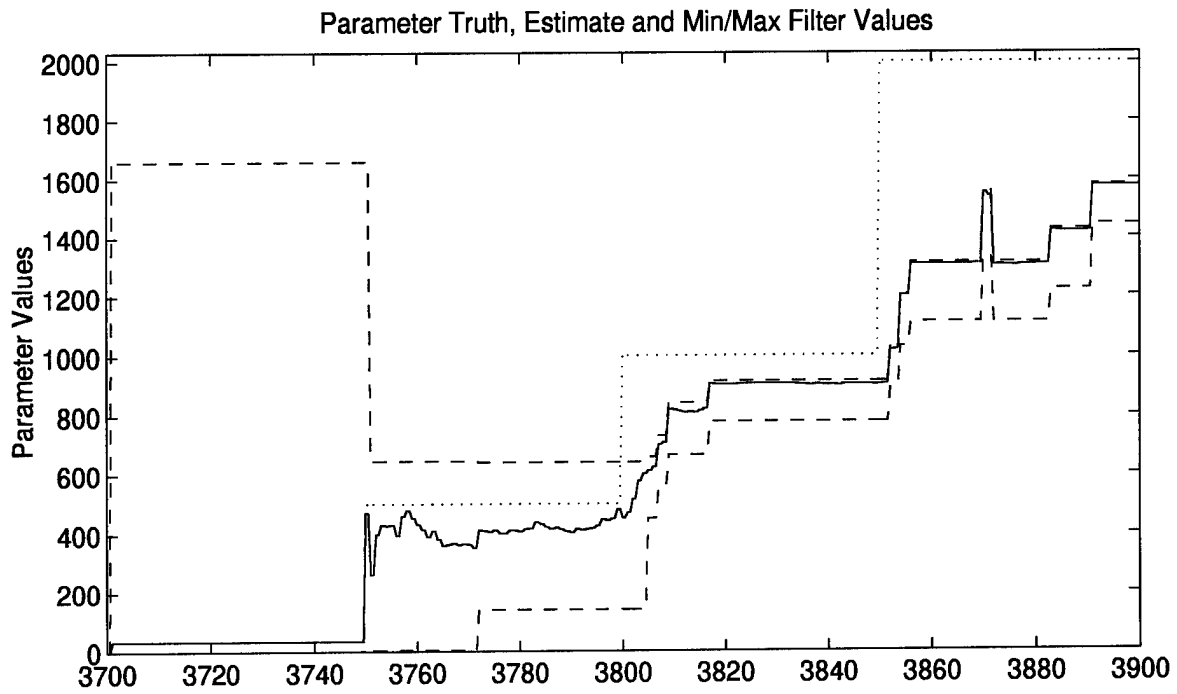


Figure 82. Parameter Estimation Performance – Case 4: Density / Sheldon Alg.

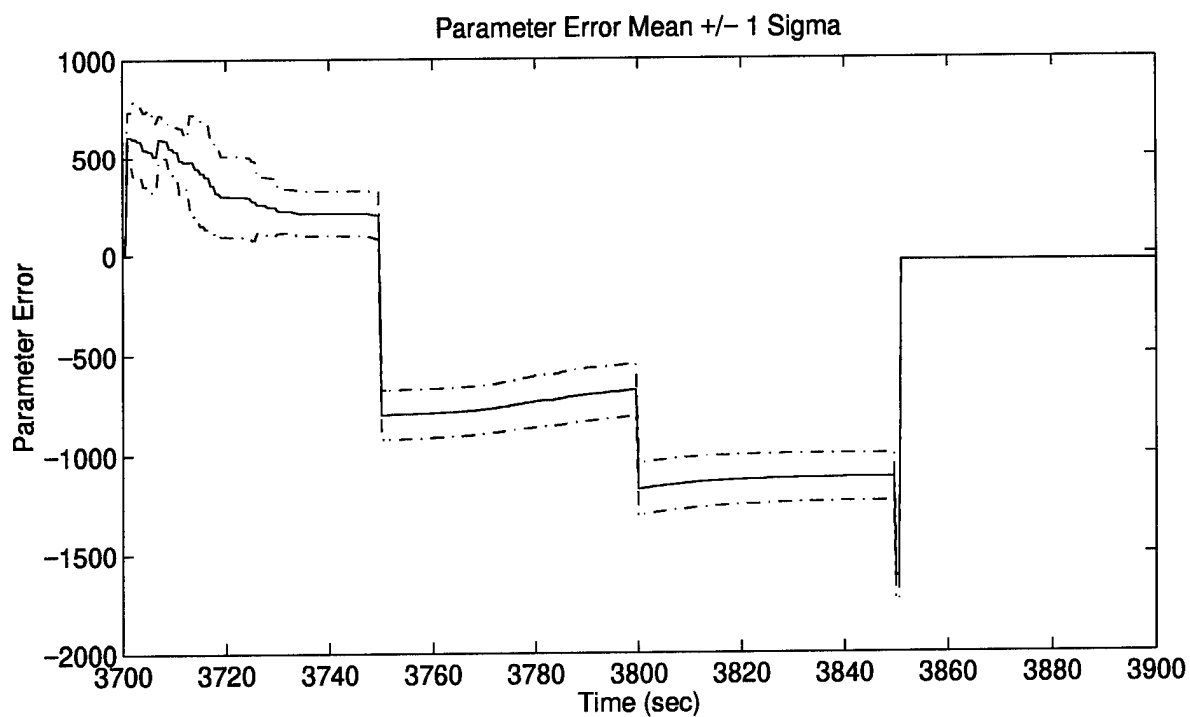
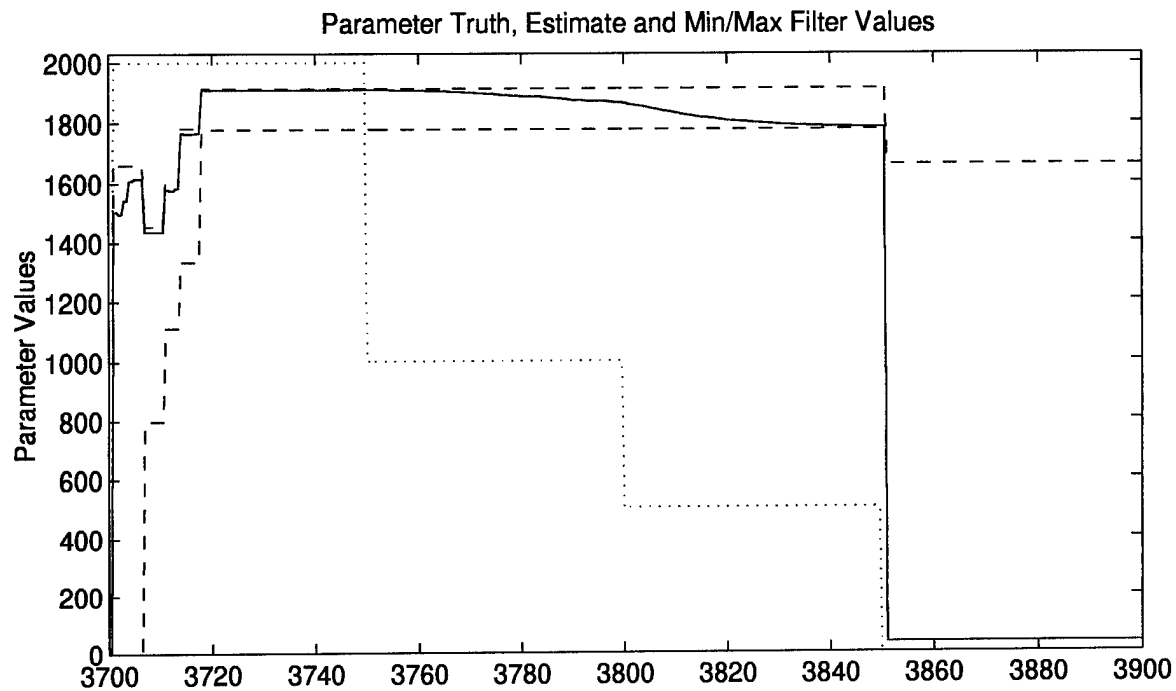


Figure 83. Parameter Estimation Performance – Case 5: Density / Sheldon Alg.

First, and somewhat less significant, is the biased low parameter estimate prior to $t = 3750$ sec when the interference/jamming level decreases. This is again caused by the Sheldon discretization choosing a maximum filter-assumed parameter value of $a_5 = 1913$, which is less than the right endpoint, $a_{\max} = 2000$, passed by the density algorithm. Furthermore, since $a_t = 2000$, the blended parameter estimate is guaranteed to be less than truth.

Second, as the bank continually contracts due to the movements and *not* from a normal contraction decision, the filters' residuals become less distinguishable. This becomes a problem when the interference/jamming level decreases at $t = 3750$ sec, and a move left is desired but not made. The lack of filter residual distinguishability prevents measure M_5 in Equation (86) from crossing threshold T_1 and allowing a move left as depicted by the logic in Figure 21. This lack of distinguishability is further illustrated in Figure 84 by the extremely slow transfer of the probability from filter 5 to filter 1 during the time frame of $t = 3750$ sec to 3850 sec. An expansion decision would be useful here but is currently suppressed. The next section will show the benefit of allowing expansions for this case. The extreme overestimation of the true parameter value results in overly conservative tuning of states 1 and 2, as shown in Figures 85 and 86.

4.8.3 Performance *With* Expansion and Additional Delay

Case 1 : The ability to expand the MMAE filter bank will become more useful in case 5, but its negative characteristics are highlighted here in case 1. Observation of the parameter plot in Figure 87 shows the somewhat erratic behavior of the moving-bank algorithm. For example, a series of [soft move right - expand - soft move right - expand] occurs from $t = 3807$ sec to 3825 sec, even though the true parameter value is unchanged. In fact, the probability plot in Figure 88 shows that filter 5 is identified as the best match to truth (it has the majority of the probability) during this entire time frame. This is also illustrated in Figure 87 since the parameter estimate trace (—) overlays the

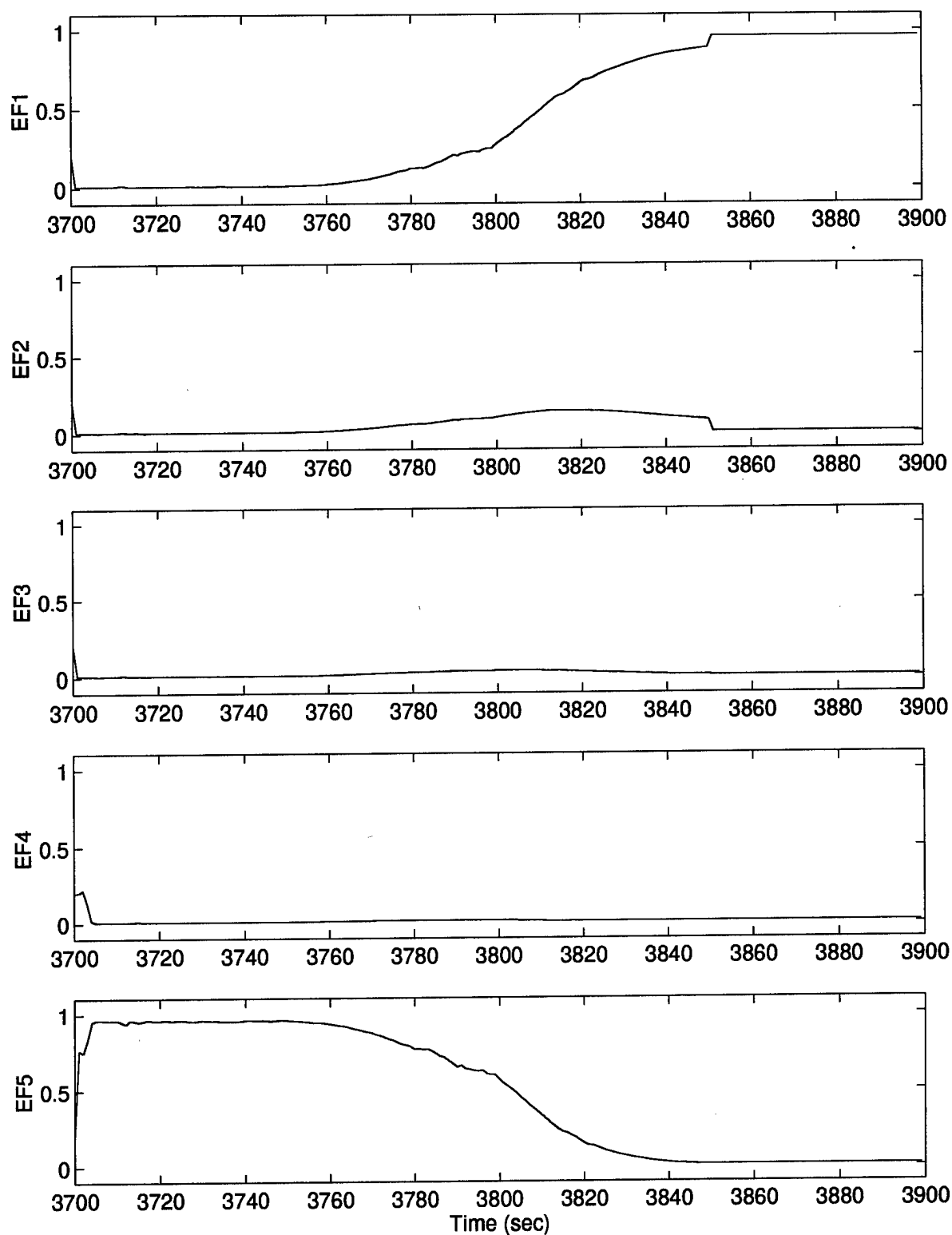


Figure 84. Elemental Filter Probabilities – Case 5: Density / Sheldon Alg.

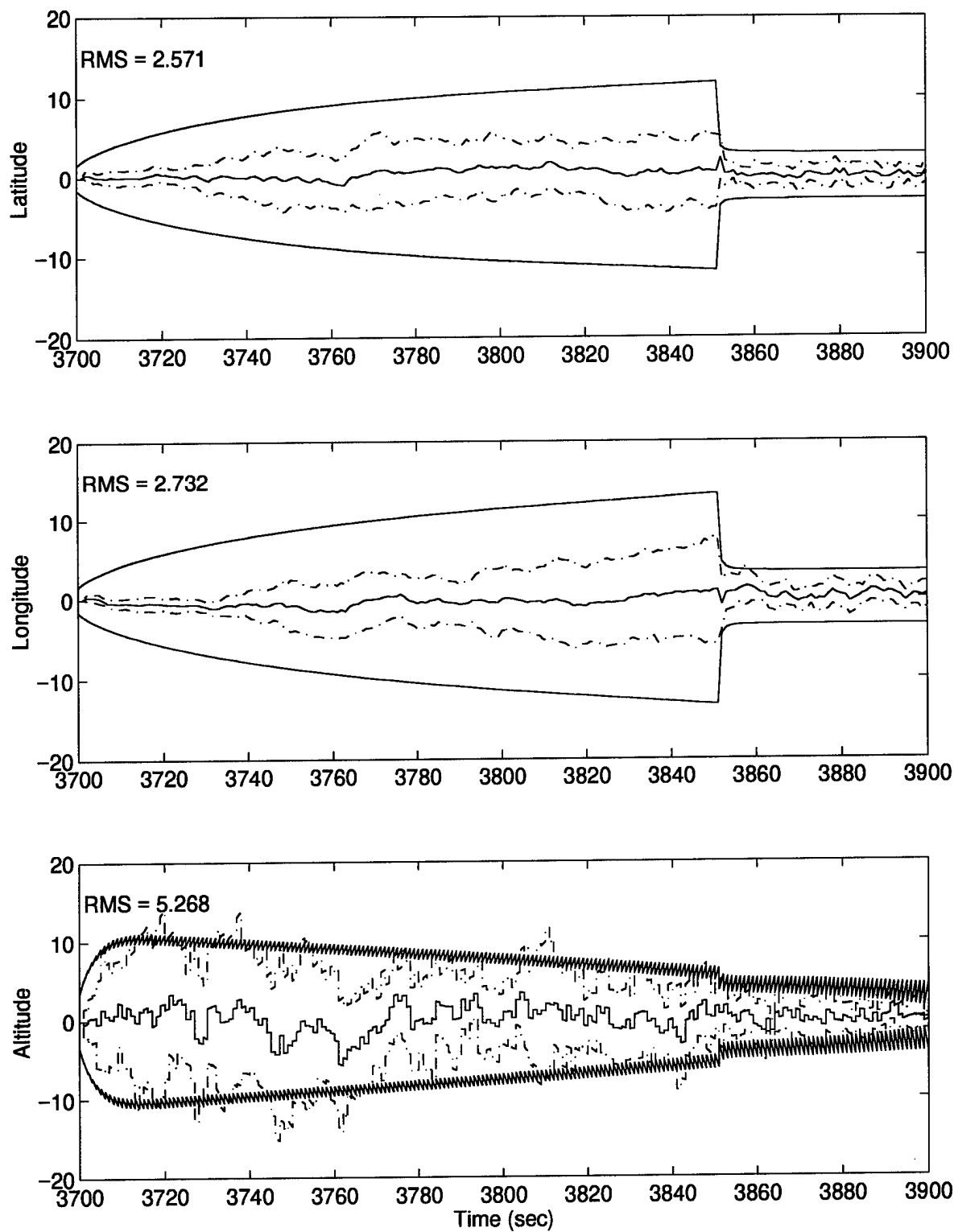


Figure 85. MMAE State Estimation Errors (feet) – Case 5: Density / Sheldon Alg.

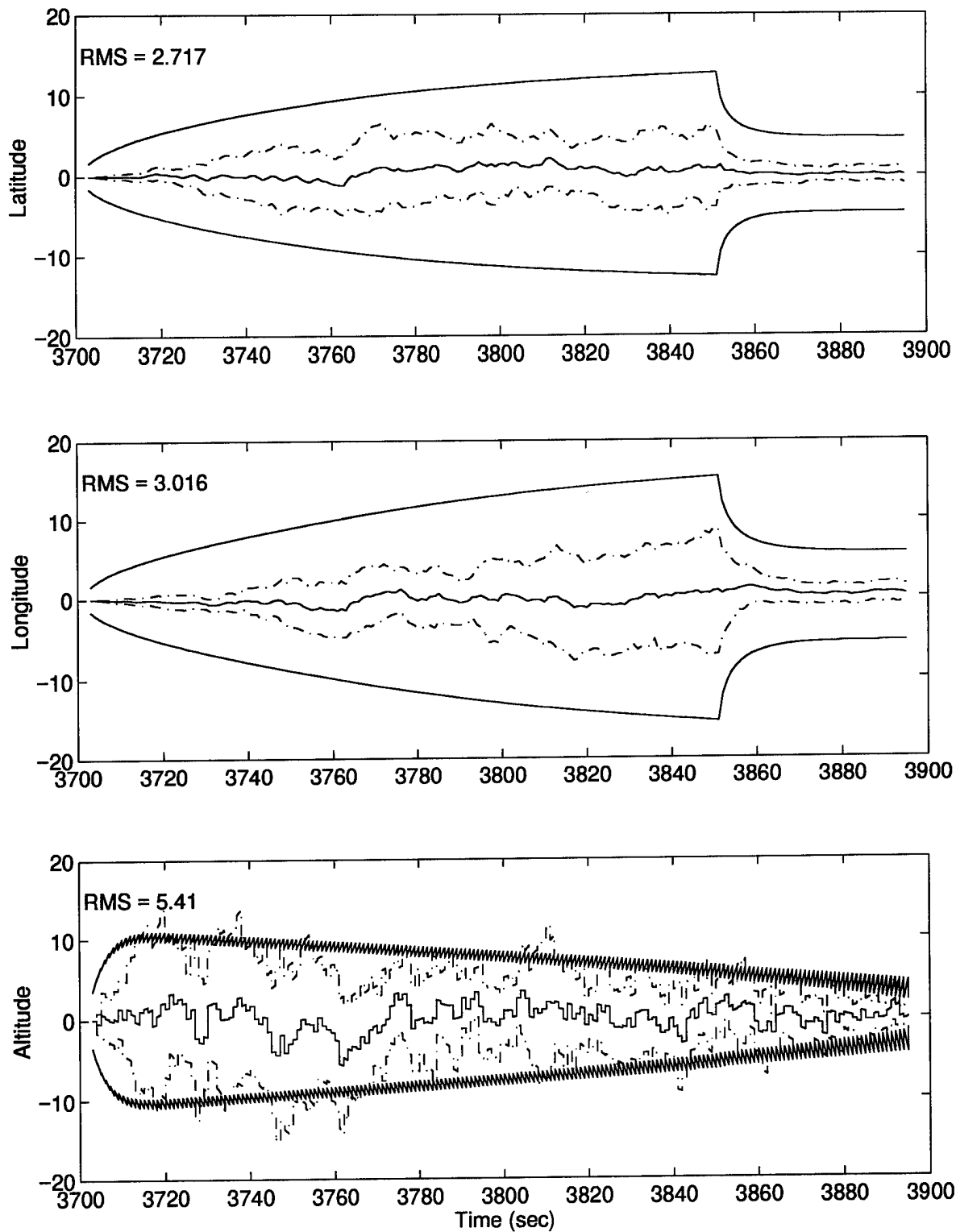


Figure 86. M³AE State Estimation Errors (feet) – Case 5: Density / Sheldon Alg.

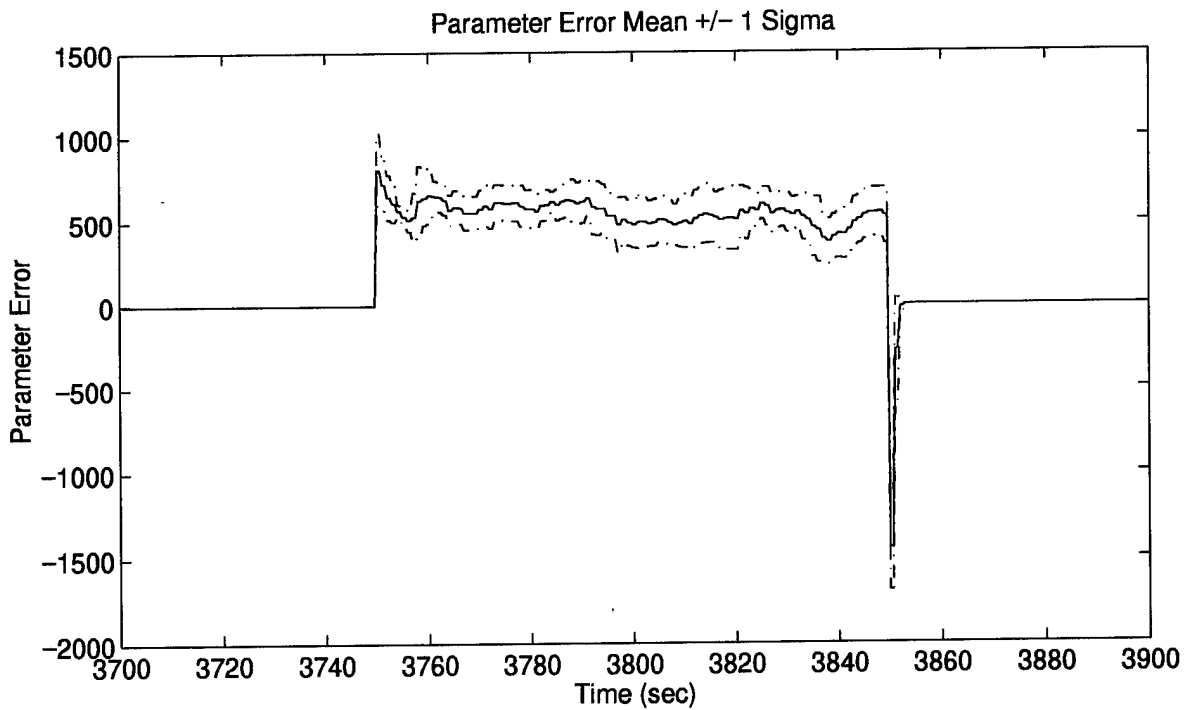
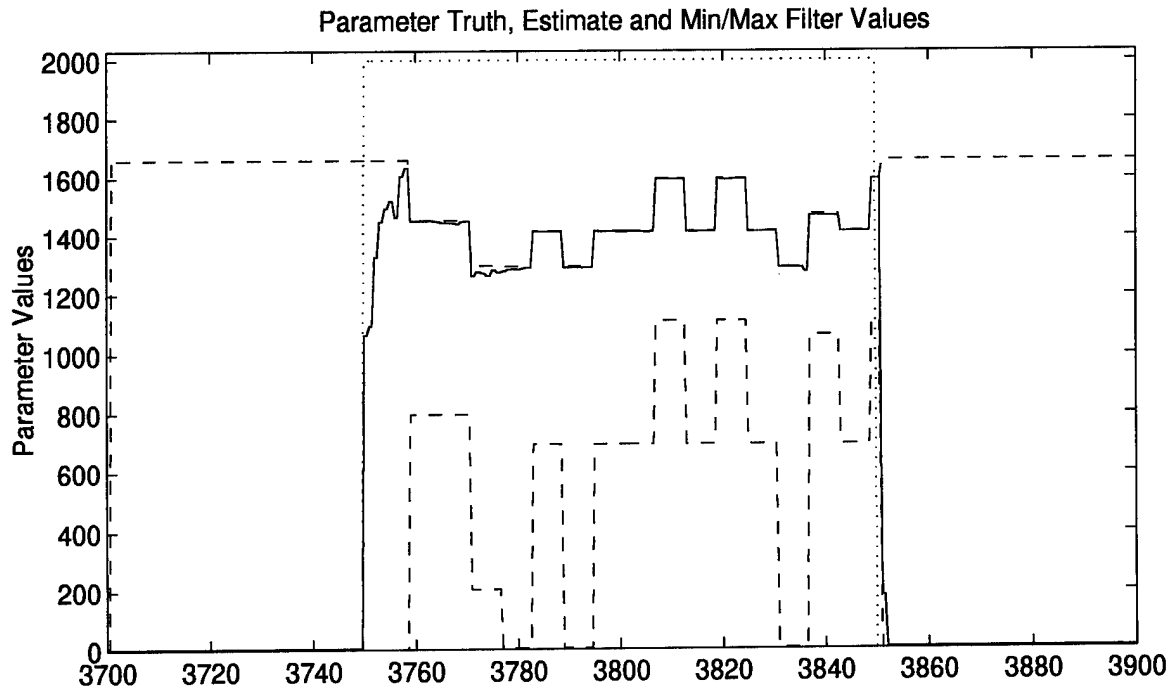


Figure 87. Parameter Estimation Performance – Case 1: Density / Sheldon / Expansions / Delay

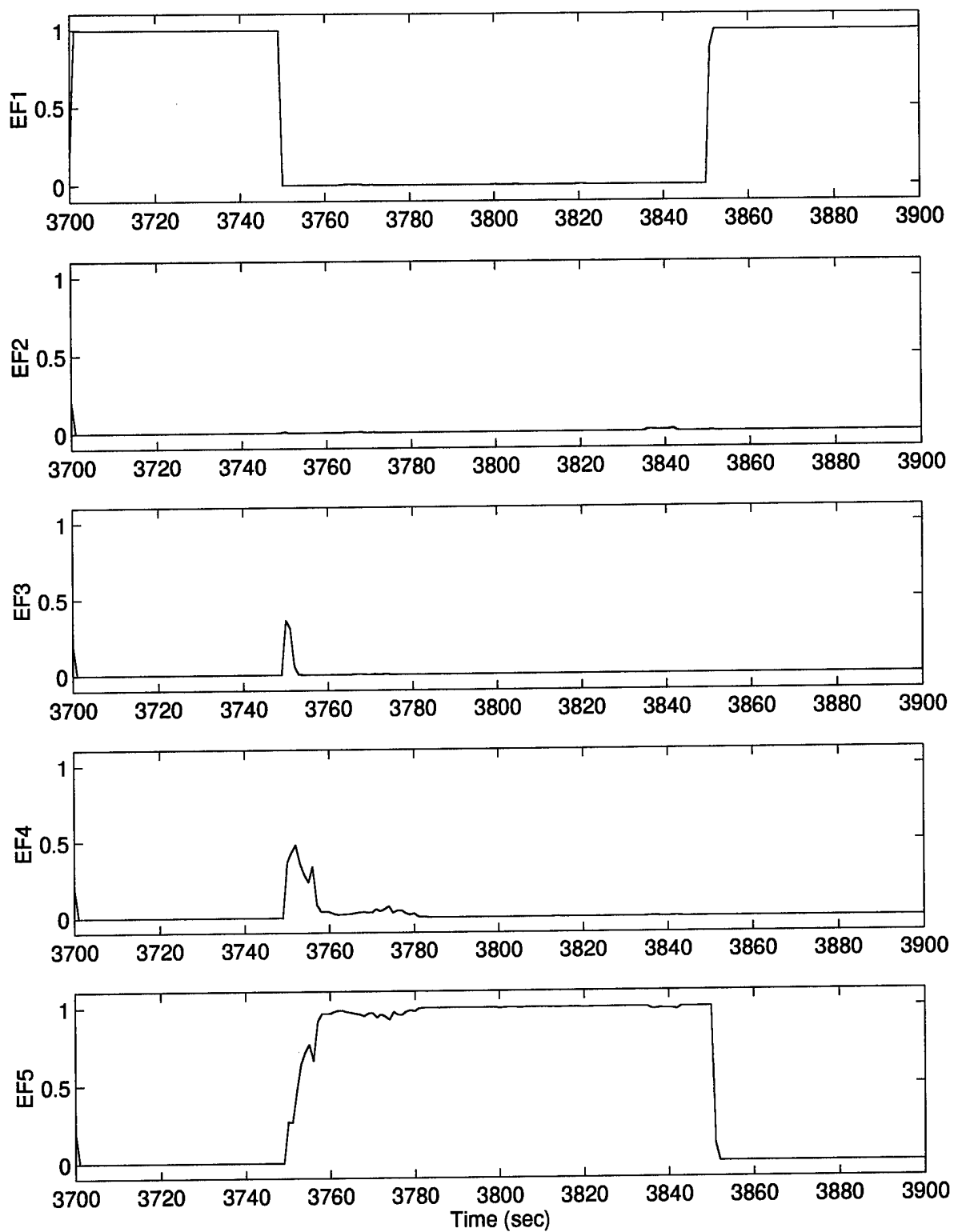


Figure 88. Elemental Filter Probabilities – Case 1: Density / Sheldon Alg. / Expansions / Delay
230

maximum filter-assumed parameter trace (- -) during this time frame. This erratic bank movement causes degradations in both the state and parameter estimates when compared to the performance of the density algorithm/Sheldon discretization *without* expansions (see measures \hat{e}_{RMS}^x and \hat{e}_{RMS}^a in Tables 16 – 22 in Appendix E).

Of greater importance, is the severe underestimation of the parameter that exists in Figure 87. Again, the Sheldon discretization selects the largest filter-assumed parameter value, a_5 , less than the maximum admissible parameter value, a_{max} , resulting in an upper bound on the parameter estimate that is considerably less than truth. To compound the problem, the expansion process is causing the parameter estimate to move away (to the left) from truth by reducing the largest filter-assumed parameter value. This is poor choice in light of the observation made in the previous paragraph, that filter 5 is identified as the best match to truth, indicating a need to move right. The decision logic tries to compensate by invoking a move right, but the expansion that follows simply moves the bank back to the left. This is best illustrated from $t = 3780$ sec through 3830 sec. Therefore, a shortcoming in the current logic needs to be corrected. One simple method would be to check which filter has the maximum probability, and if this filter is at an edge of the current bank (i.e. a_1 or a_J), then the expansion process must not be allowed to move the bank away from this filter. For the example shown in here in case 1, the bank would not be allowed to move left *away* from a_J .

Further notice that Figures 87 and 78 are *significantly* worse than Figure 57, in which the bank right edge *was* allowed to go all the way to $a_t = 2000$. By so doing, the parameter estimate is given the latitude to become much more accurate. With $a_t = 2000$, we again would want to allow the modified probability blending of Section 4.11 to remove artificially biased estimates due to the impact of lower bounds—though of less relative importance at $a_t = 2000$ than at $a_t = 1$. Finally, case 2 will be discussed presently and shows that this artificial degradation of the parameter estimates is

removed entirely when the bank is allowed to span the range including the true parameter value, by allowing bank motion over the *entire* admissible region of the parameter space.

Although time did not permit implementation of the modification to the expansion logic or the modified blending for these test cases, an additional simulation was conducted to illustrate the benefit of allowing the largest filter-assumed parameter value to exceed the true parameter value. This case was only simulated for this algorithm (MMAE incorporating the density algorithm with Sheldon discretization *with* expansions and additional delay), and the parameter estimate performance is shown in Figure 89. This case is identified as case 7 with a GPS noise interference level of $a_t = 1500$ for $3750 \text{ sec} \leq t \leq 3850 \text{ sec}$. Notice that the parameter underestimation has been significantly reduced. Comparison of the parameter error mean values in Figures 87 and 89 is further evidence that the parameter estimate is no longer suffering from a *large* bias. The significant improvement in performance shown here is strong motivation for allowing the bank motion to cover the entire admissible parameter space, even though the Sheldon discretization (as implemented) would prevent this from occurring. Again, Chapter 5 states the recommendation to pursue well-developed methods that allow this type of bank motion.

Returning to case 1, the error state estimate plots do not reveal any additional insights and are very similar to the plots shown earlier for the density algorithm *without* expansions, so they are not presented. The primary indicator of performance between these algorithms is \hat{e}_{RMS}^a .

Case 2: The biasing problem encountered with the combined density algorithm/Sheldon discretization for this case is partially reduced here via bank expansions. Observe the bias high in Figure 90 at $t = 3770 \text{ sec}$ through 3775 sec . This is later rectified by an expansion at $t = 3776 \text{ sec}$, which effectively changes the parameter discretization within the MMAE bank such that filter 4, which is a good match to truth, obtains the greatest probability weight, as shown in Figure 91. This analysis is not meant to imply that simply adding expansion decisions will *ensure* the selection by

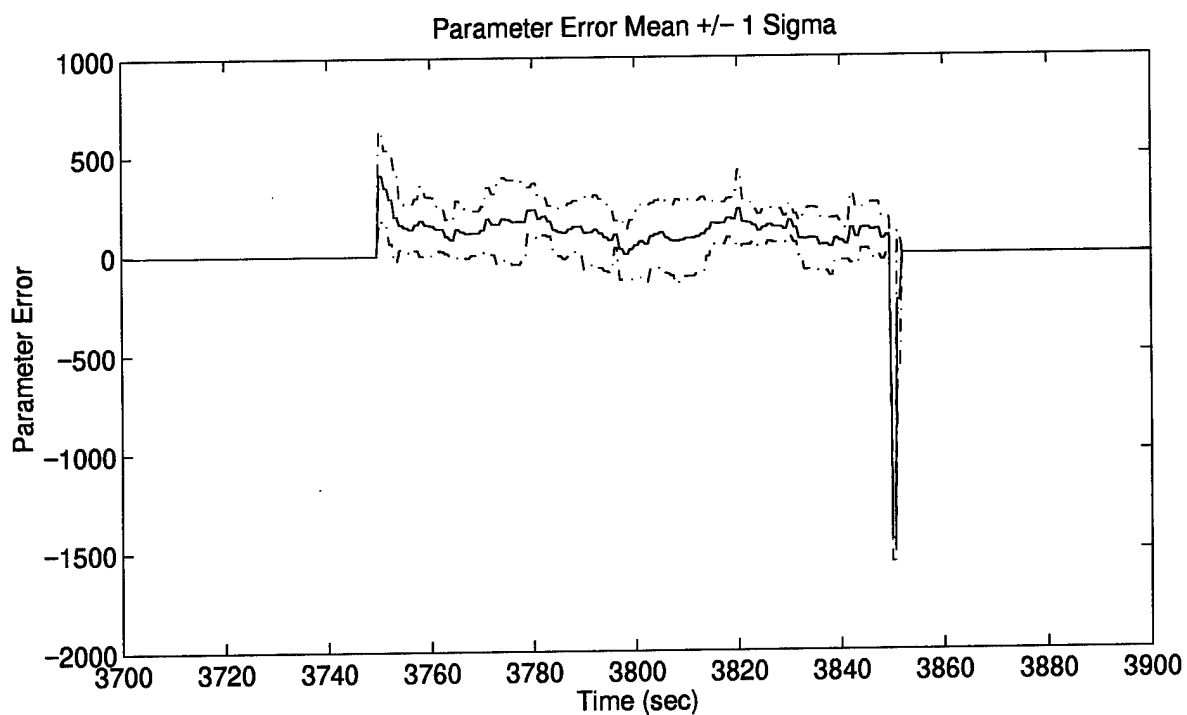
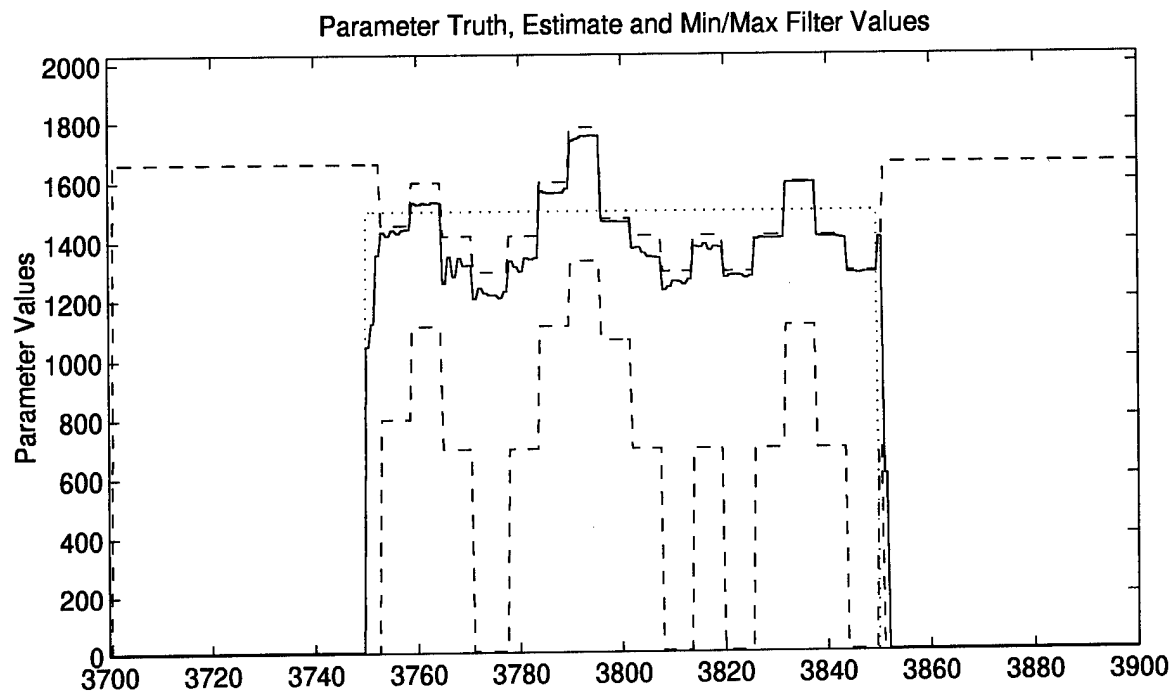


Figure 89. Parameter Estimation Performance – Case 7: Density / Sheldon / Expansions / Delay

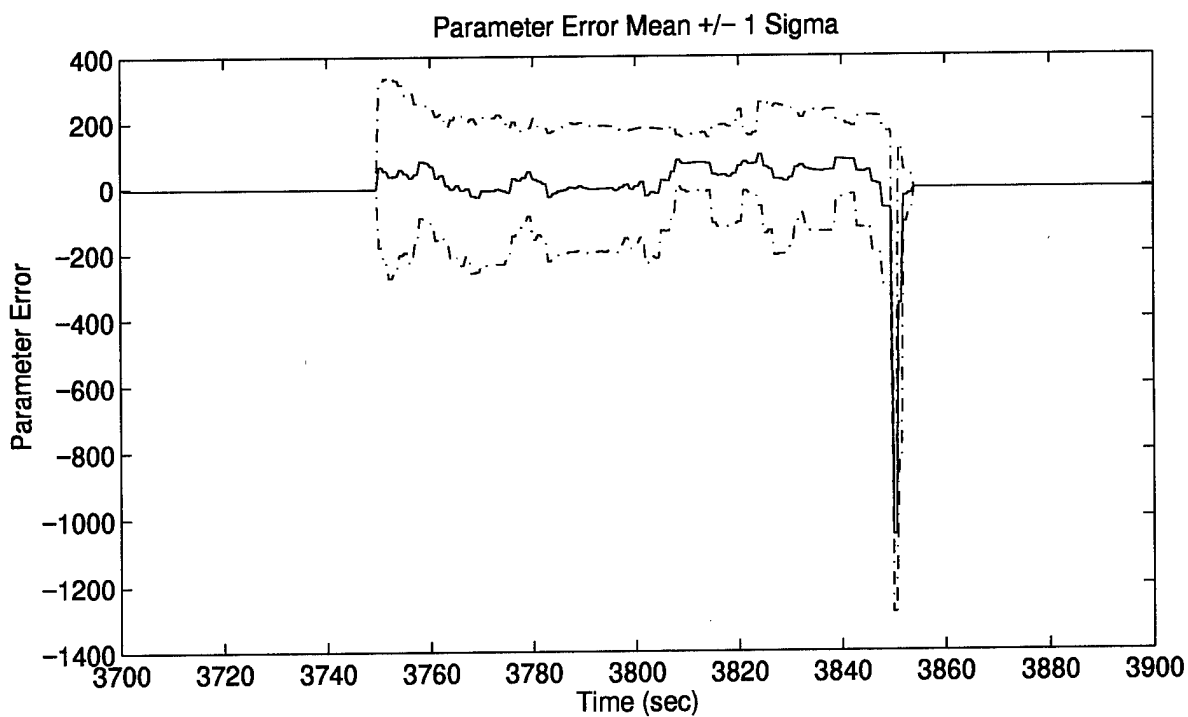
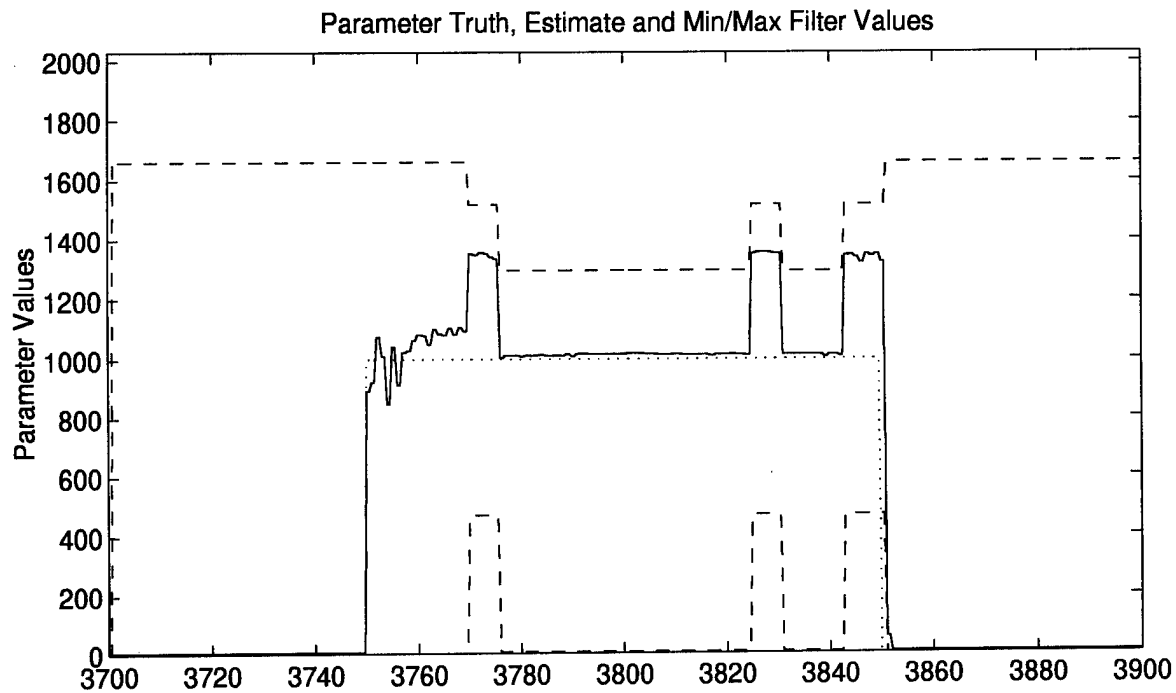


Figure 90. Parameter Estimation Performance—Case 2: Density / Sheldon / Expansions / Delay

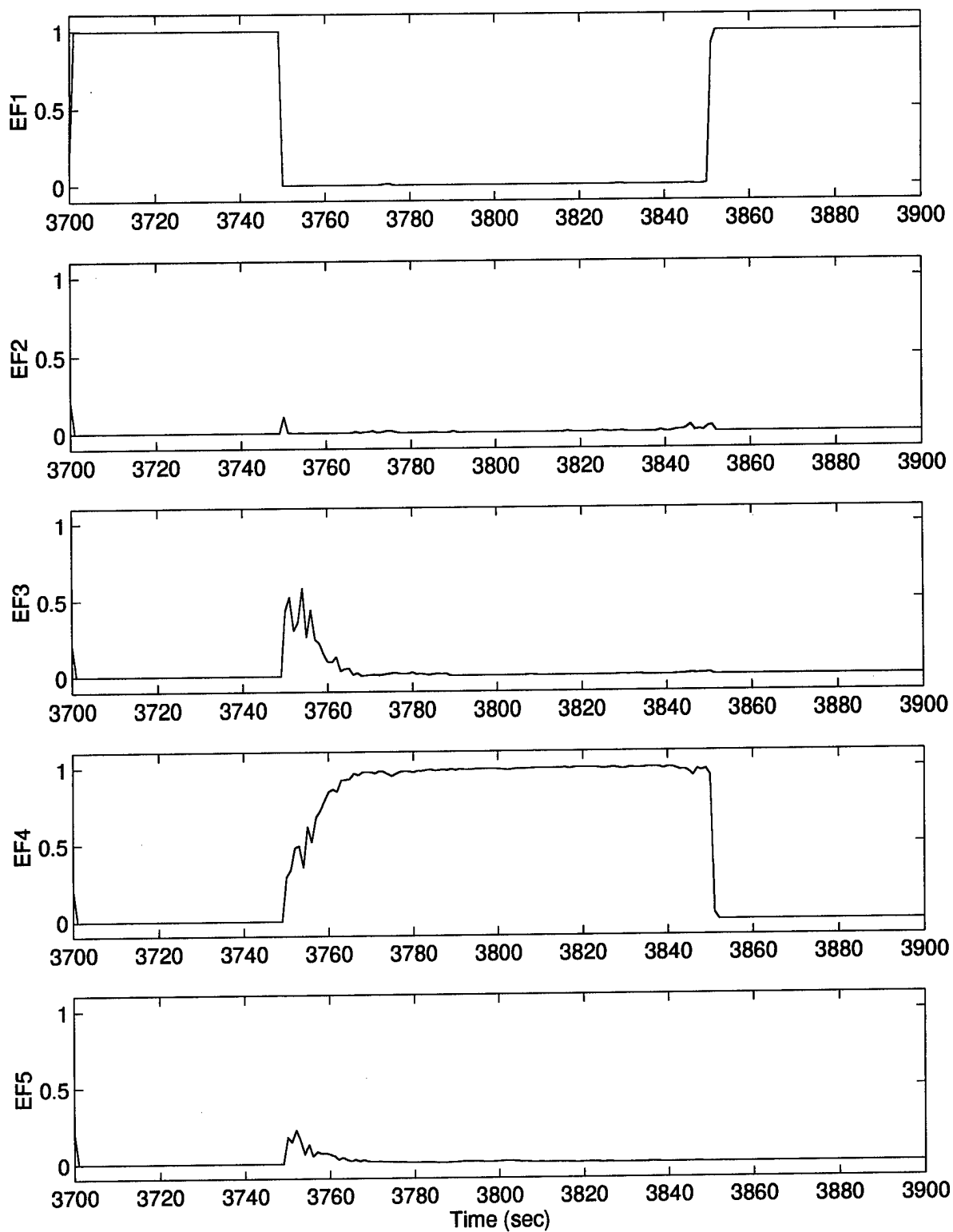


Figure 91. Elemental Filter Probabilities – Case 2: Density / Sheldon / Expansions / Delay
235

the Sheldon discretization of a filter-assumed parameter value that matches truth well. This again motivates the recommendation to take the parameter values chosen by a Sheldon discretization and set the one closest to \hat{a}_{MMAE} equal to \hat{a}_{MMAE} .

Case 3: The same analysis presented under case 3 for the density algorithm/Sheldon discretization *without* expansions and additional delay applies here, and the performance plots are not presented.

Case 4: The same analysis presented under case 4 for the density algorithm/Sheldon discretization *without* expansions and additional delay applies here, and again the performance plots do not warrant being shown.

Case 5: Recall the need to allow expansions discussed under case 5 for the density algorithm/Sheldon discretization *without* expansions and additional delay. Comparison of the parameter estimates shown in Figures 83 (no expansions) and 92 (with expansions) indicates the pros and cons of adding expansions to this algorithm.

The primary benefit is the ability to respond to a decrease in the interference/jamming level at $t = 3750$ sec by expanding the bank and ensuring the filters' residuals are distinguishable. Without such expansions, the movement-induced contractions cause the filter's residuals to become indistinguishable and prevent needed bank movements. This problem is not completely resolved, as seen by the series of soft moves left from $t = 3782$ sec to 3800 sec, which consequently contract the bank, resulting in a bank which is too finely discretized and lacking in filter residual distinguishability. This point is further supported by observing the slowly changing probabilities for filters 1 and 5 shown in Figure 93 at $t = 3800$ sec to 3850 sec. The problem encountered with allowing expansions is erratic moving-bank decisions, resulting in poor parameter tracking, as seen by the numerous soft moves right and expands from $t = 3707$ sec to 3749 sec.

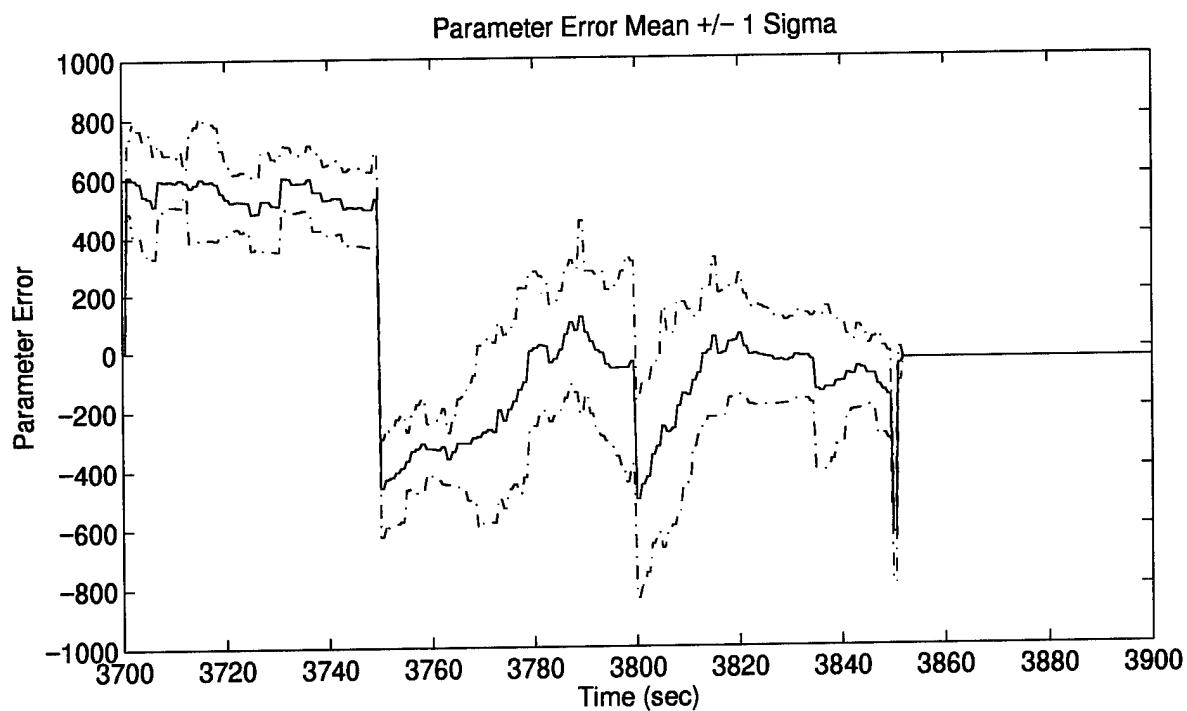
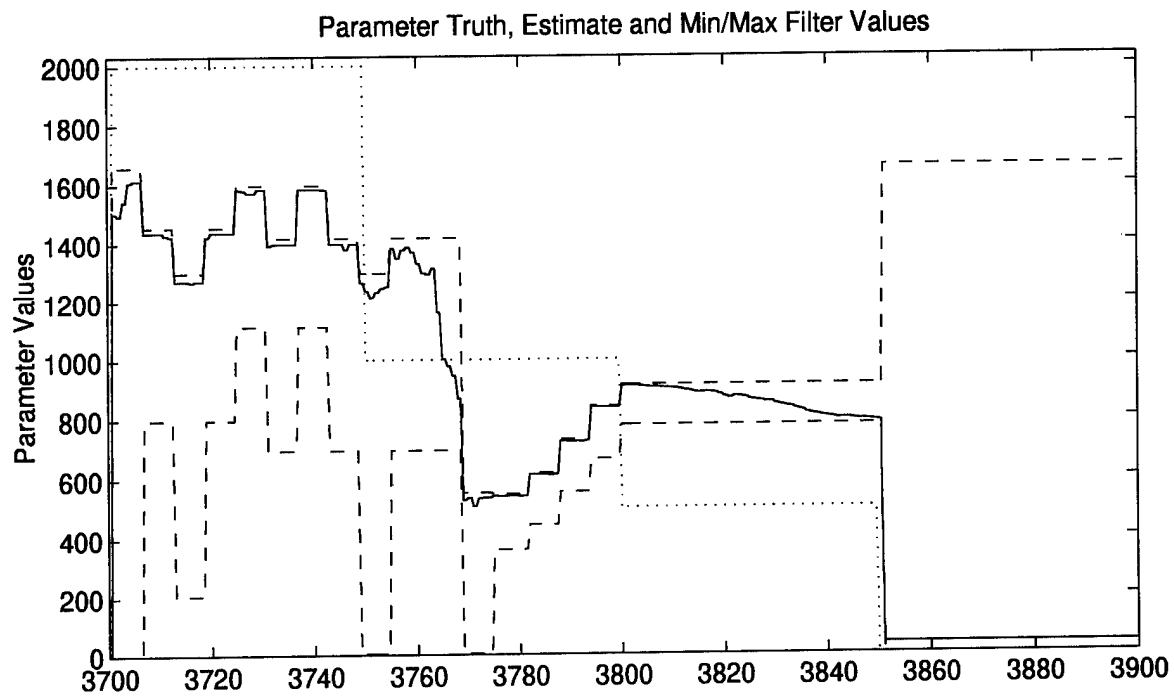


Figure 92. Parameter Estimation Performance – Case 5: Density / Sheldon / Expansions / Delay

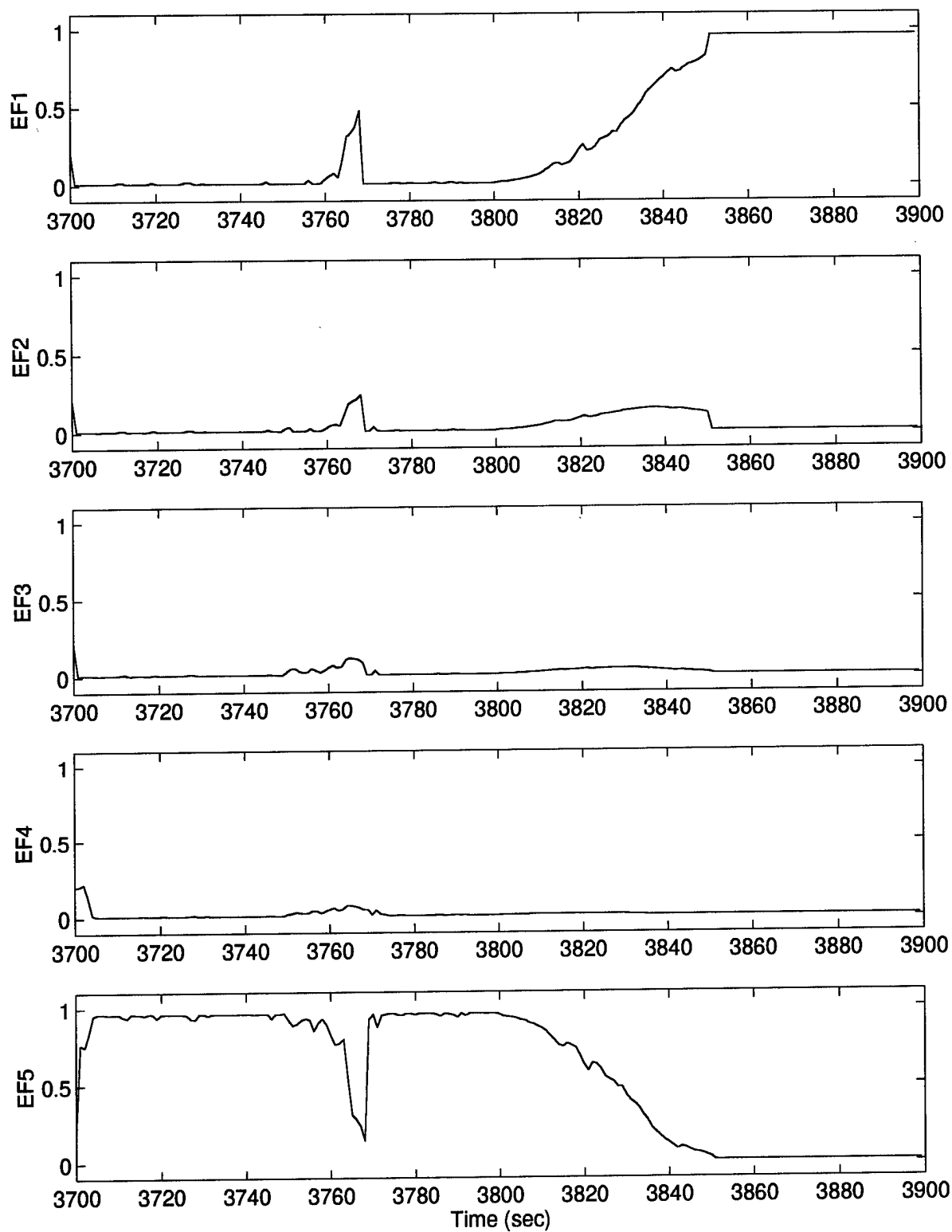


Figure 93. Elemental Filter Probabilities – Case 5: Density / Sheldon / Expansions / Delay
238

4.9 MMAE Incorporating the Probability Algorithm

4.9.1 Implementation Issues

The probability algorithm presented in Section 3.2 was implemented to drive the decisions for the moving-bank MMAE. Again, five elemental filters were used and the probability-based discretization method generated a look-up table of parameter values for on-line use. The probability values used by the PBDM are shown in Table 7. Note that $P_{\chi_3} = 0.5$ and with $m = 6$ for this problem,

Table 7. Chi-Squared Probability Values

Filter #	1	2	3	4	5
Chi-Squared Probability Value, P_{χ_i}	0.01	0.24	0.5	0.76	0.99

the threshold is given by MATHCAD [39] as $T = qchisq(0.5, 6) = 5.348$. The threshold, T , is first defined in Section 3.2, and the MATHCAD approach is explained in Section 3.2.3. A manual search for filter-assumed parameter values satisfying the probabilities in Table 7 was conducted *once* to determine M_{eig}^d for filters 1, 2, 4 and 5. See Section 3.2.3.1 for the discussion of this manual search and for the definition of M_{eig}^d . A value of $a_t = 1000$ was used for the manual search since it lies close to the center of the admissible parameter space. Given more insights into the sensitivity of the system to a larger or smaller value for this parameter, a different choice for a_t might be motivated. Next, filter-assumed values of $a_j = 1, 10, 100, 500, 1250, 1750$ and 2000 were combined with $a_t = 1000$ to calculate P_{χ_j} via numerical integration, giving the designer an idea of which filter-assumed values would result in the desired values for P_{χ_j} . Finally, approximately 20 more choices for a_j were needed to realize the desired P_{χ_j} values, resulting in the four values of M_{eig}^d shown in Table 8.

Table 8. Search Criteria

Filter #	1	2	4	5
M_{eig}^d	51637	25	0.414	0.2656
Initial Guess Percentage, δ_l	0.05	0.4	-	-
Initial Guess Percentage, δ_h	-	-	1.3	1.5

The search routine described in Section 3.2.3.1 automated the generation of the look-up table. The desired eigenvalue measures, M_{eig}^d , and the initial guess value percentages for the search are shown for each filter in Table 8. The initial guess value percentages, δ_l and δ_h (see Section 3.2.3.1), were quickly determined through trial and error. The search routine was initially allowed to begin the search at a_{min} for filters 1 and 2 and a_{max} for filters 4 and 5. By observing the measure M_{eig} (see Section 3.2.3.1) and realizing that it did not come close to the desired measure M_{eig}^d until the guess value had reached some neighborhood of \hat{a}_{MMAE} , it was clear that starting future searches at the extreme values of a_{min} and a_{max} was inefficient. The values for δ_l and δ_h simply started future searches such that M_{eig} was in the neighborhood of M_{eig}^d . The initial guess percentages are still conservative and ensure the initial guess values $a_l = \delta_l \cdot \hat{a}_{MMAE}$ and $a_h = \delta_h \cdot \hat{a}_{MMAE}$ are sufficiently low and high respectively. The number of guesses used in the automated search (see Section 3.2.3.1) was 40 ($N_g = 40$). Larger values for N_g did not significantly change the results of the search routine but did significantly increase the computer processing time required to perform the search. A 40-by-5 look-up table of parameter values was created for on-line use. The first dimension represents the 40 discrete values assumed for the true parameter value over the range $1 \leq a_t \leq 1,950$ at an *arbitrary* interval spacing of 50. Recall that $R_{GPS} = a_t R_0$, where R_0 is the nominal noise covariance value of 9 ft^2 . The second dimension represents the need to store a look-up value for each of the five filters.

New filter initialization using \hat{x}_{MMAE} and a decision delay of 2 sample periods were implemented for enhanced performance. The probability algorithm's propensity to oscillate about the true

parameter value by making repeated moves to the left and right motivated an increase in the decision delay from 1 to 2 sample periods. Additionally, a dead zone in the form of a minimum move size was used to counter unwanted movements. Recall the guideline given in Section 3.2.5 to set the minimum move size equal to one-third the average distance from the table-stored bank centers to their endpoints. This led to a minimum move size = $900/3 = 300$. Finally, the practice of lower bounding the probabilities was implemented with the previously used value of $p_{\min} = 0.001$.

4.9.2 Performance

Case 1: The plot of the parameter estimate shown in Figure 94 indicates the relatively good tracking ability of this algorithm. A short convergence time of 5 to 6 samples is required for the algorithm to adapt to the onset of interference/jamming at $t = 3750$ sec and reach a parameter estimate value very close to truth. Notice a gradual increase in the minimum filter-assumed parameter value (---) from $t = 3752$ sec to 3762 sec, resulting from the gradual convergence of \hat{a}_{MMAE} towards a_t via parameter position estimate monitoring. Similarly, when the interference/jamming is turned off, a gradual decrease in the minimum and maximum filter-assumed parameter values is seen from $t = 3852$ sec to 3862 sec. A momentary drop in the parameter estimate at $t = 3820$ sec results from an unusually large noise sample. The wide parameter breadth of the bank ($a_J - a_1$) permits the parameter estimate to drop significantly, unlike the algorithms incorporating the density algorithm, which did not suffer such a dramatic change in the parameter estimate at this time. This demonstrates a potential liability of allowing the bank to maintain a broad or peripheral view of the parameter space. The same undesirable transient behavior described under case 1 of the fixed-bank performance is seen here once the interference/jamming is removed. See the blended estimates in Figure 95 from $t = 3850$ sec to 3865 sec.

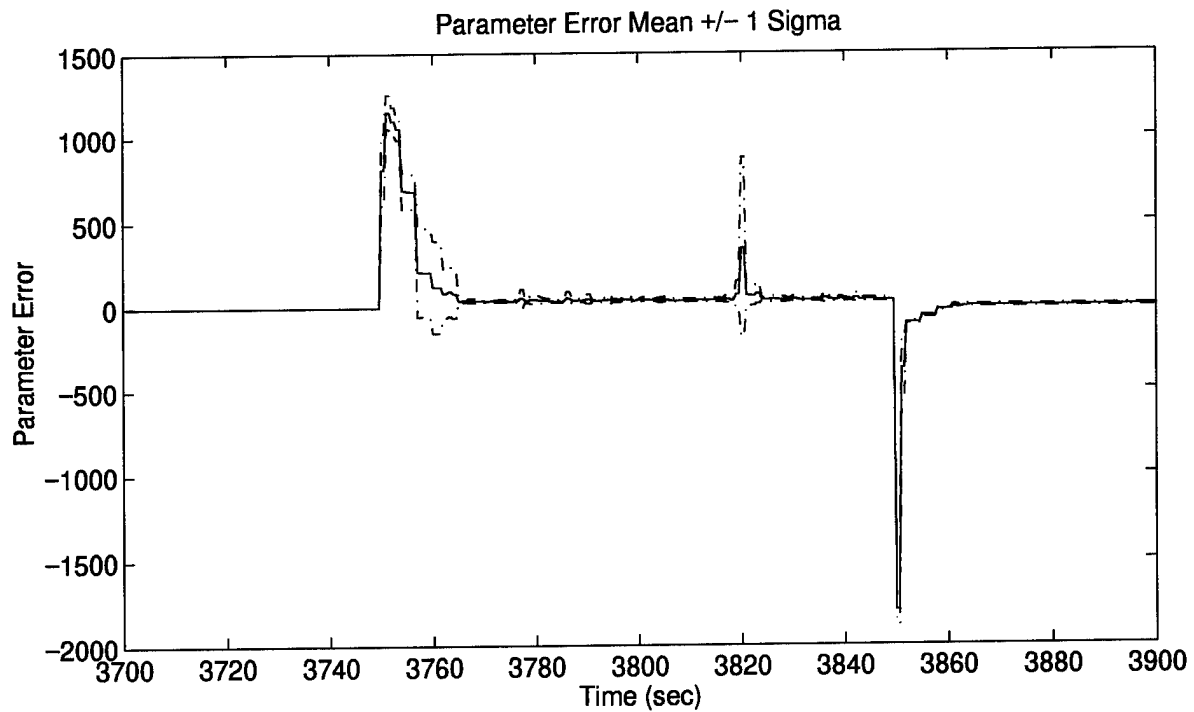
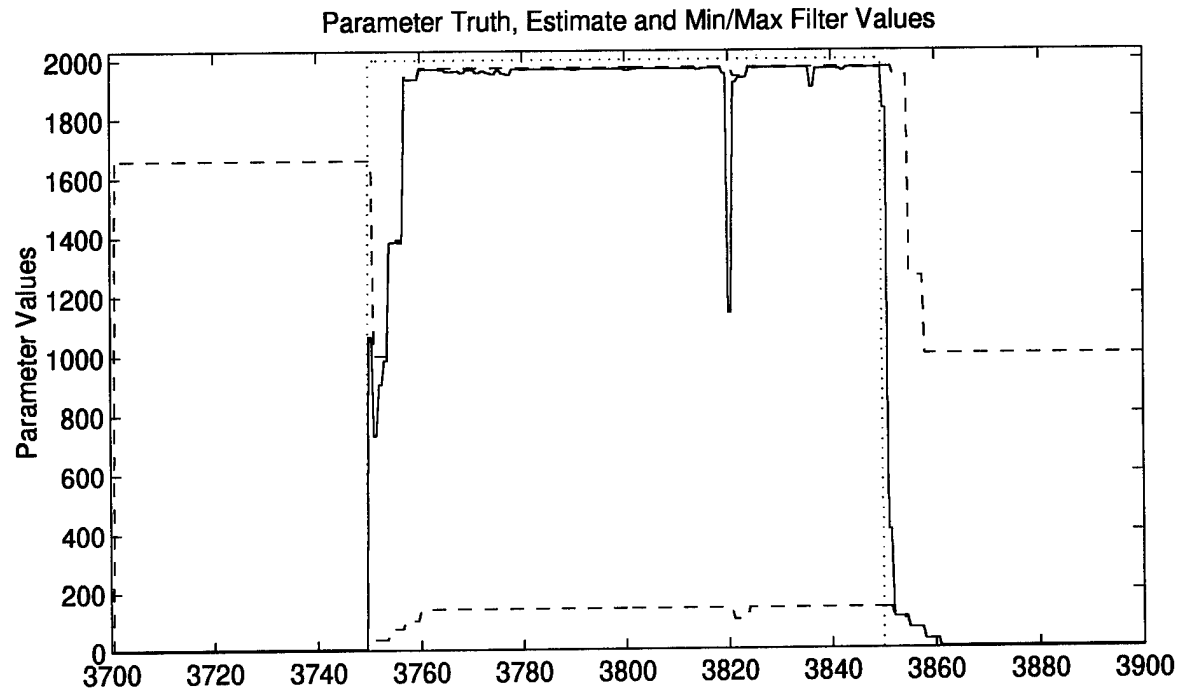


Figure 94. Parameter Estimation Performance – Case 1: Probability Algorithm

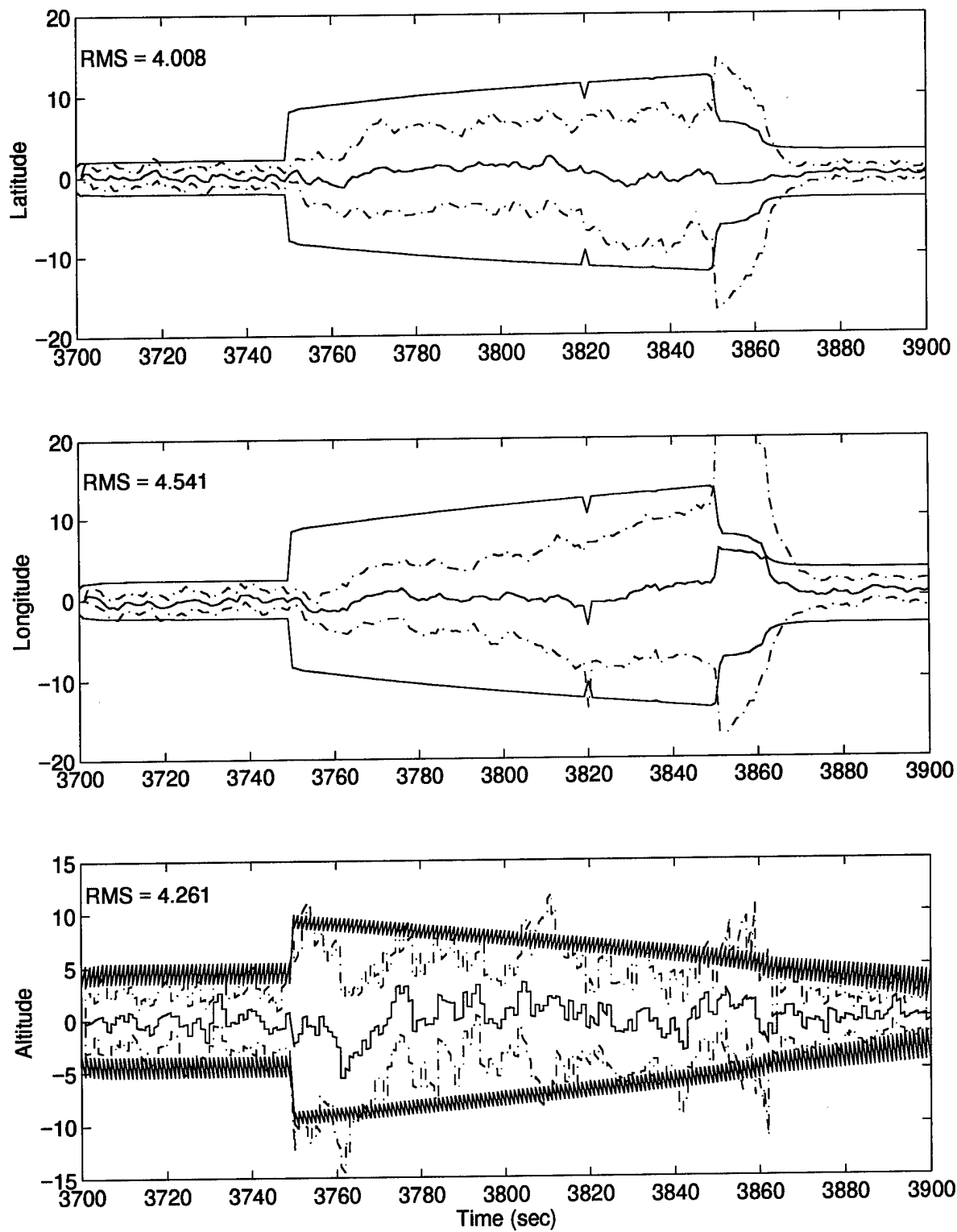


Figure 95. MMAE State Estimation Errors (feet) – Case 1: Probability Algorithm

Comparison of Figure 41 to Figure 95, along with their associated \hat{e}_{RMS}^x values, implies the superior performance of the fixed-bank algorithm over the probability algorithm for the *conventional* MMAE. However, Figures 44 and 96, along with their associated \hat{e}_{RMS}^x values, imply the superior performance of the probability algorithm over the fixed-bank algorithm for the M³AE. This supports the statement that tuning the MMAE for better parameter estimation, as with the probability algorithm, often degrades the MMAE state estimation and also provides an impetus for utilizing the M³AE architecture.

Case 2: The minimum and maximum filter-assumed parameter values are indicated by the trace pair (---) in Figure 97 and illustrate the capability of this algorithm to bound the true parameter extremely well for $t = 3750$ sec to 3820 sec. Consequently, the parameter is tracked extremely well, but this performance is not consistent, as shown by the erratic parameter estimate for $t = 3820$ sec through 3850 sec. In particular, the parameter estimate goes from severe underestimation to overshooting the true parameter value for $t = 3820$ sec through 3830 sec. The parameter estimate starts to improve well from about $t=3833$ sec to $t = 3840$ sec along with a contraction, but then again suffers severe overestimation along with too large a bank size from $t = 3842$ sec through 3850 sec. The problem can be attributed to both the moving-bank method (parameter position estimate monitoring) and the discretization method (PBDM). The parameter position estimate monitoring causes continual movement away from the true parameter value and to the right (higher parameter values). This is illustrated in Figure 97 by the parameter estimate trace (—) being superimposed on the maximum filter-assumed parameter value trace (- - -) shortly after $t = 3820$ sec and $t = 3840$ sec. Recall that the bank is centered on \hat{a}_{MMAE} , so the center of the bank is continually moving to the right. Given a better method to estimate the true parameter position, these unwanted movements could be avoided.

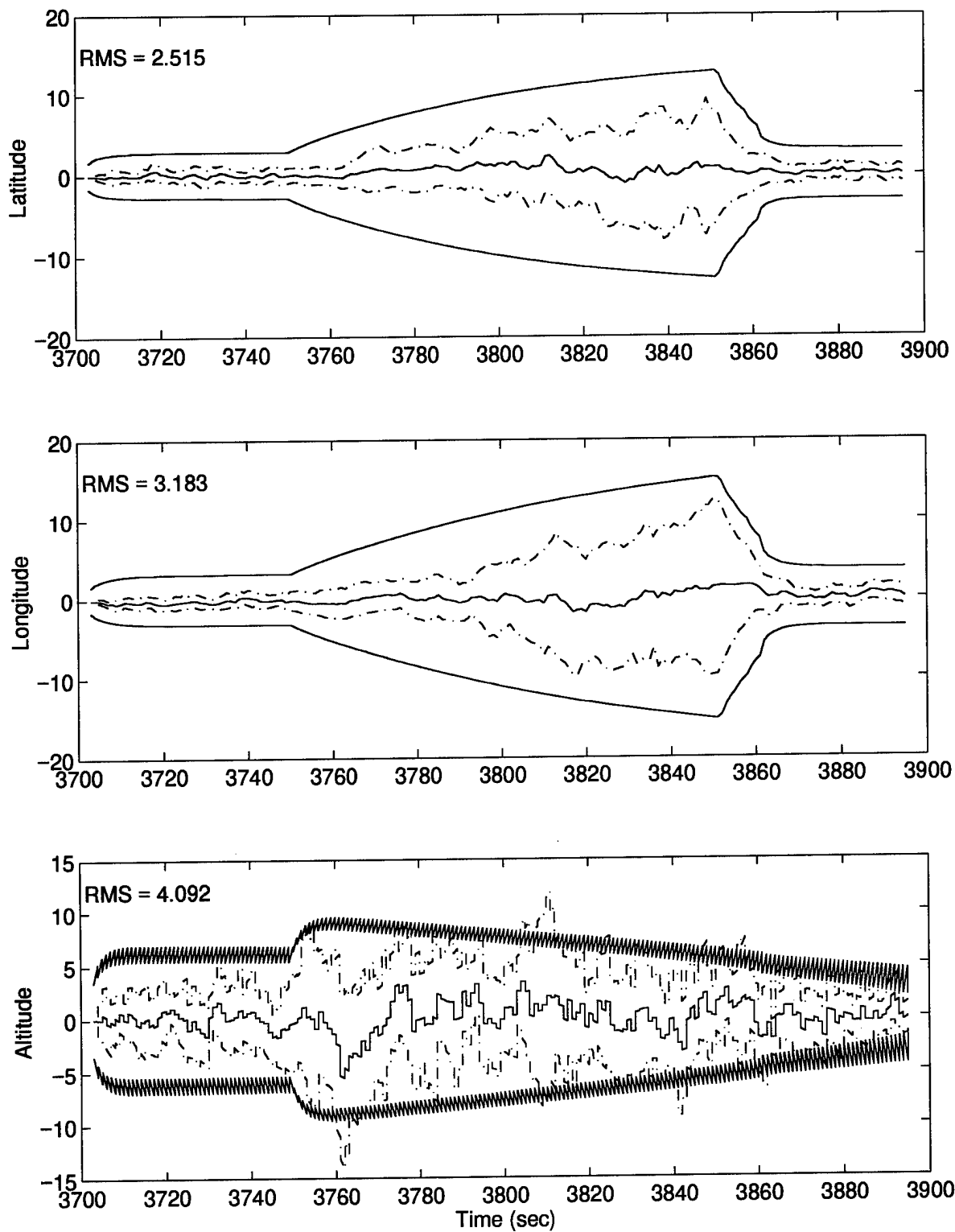


Figure 96. M³AE State Estimation Errors (feet) – Case 1: Probability Algorithm

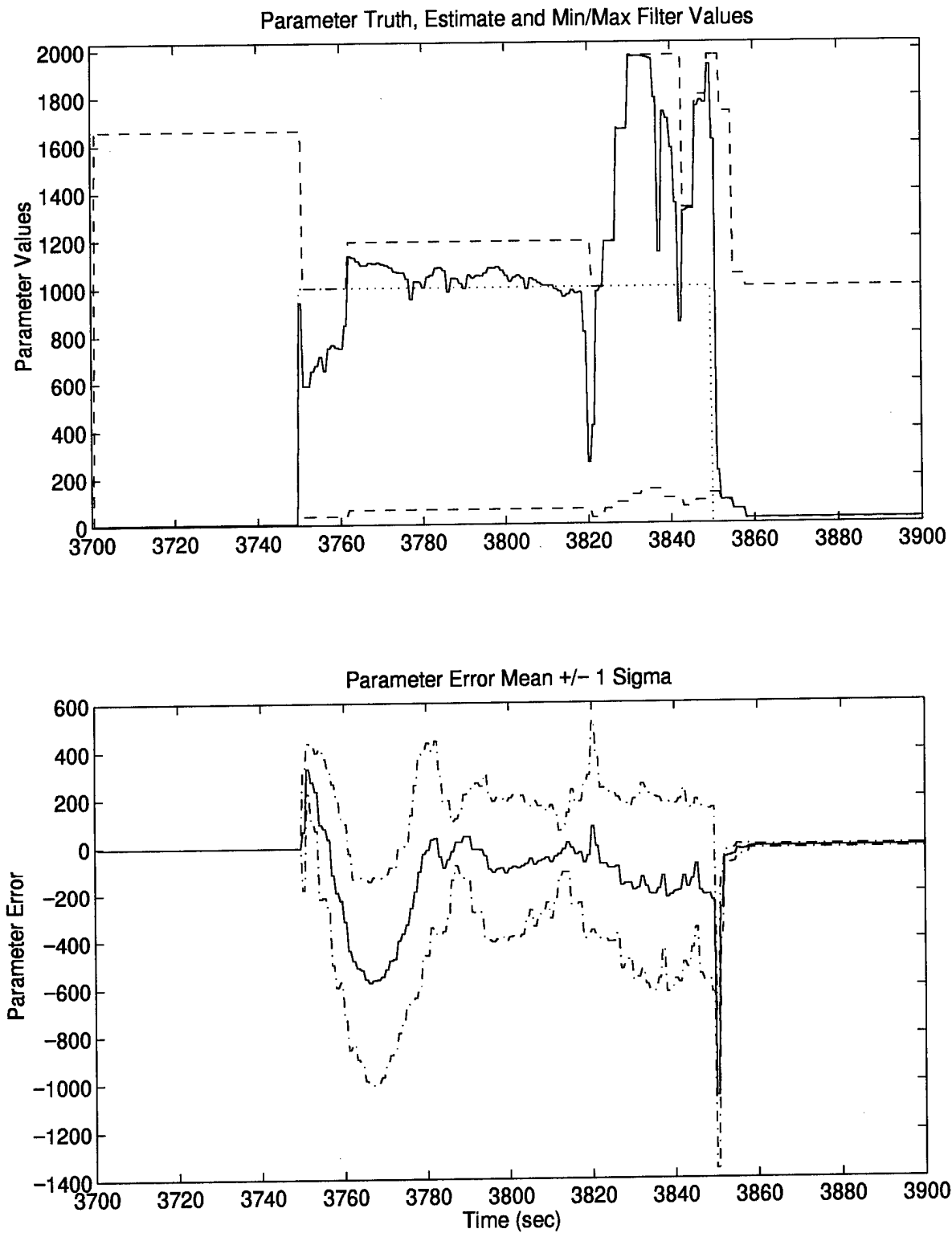


Figure 97. Parameter Estimation Performance – Case 2: Probability Algorithm

To make things worse, the PBDM assigns parameter values associated with the movements above that gradually widens the breadth of the bank. Observe the series of bank movements from $t = 3823$ sec to 3827 sec and notice that the minimum filter-assumed parameter value moves slightly to the right for each move. In contrast, the maximum filter-assumed parameter value moves quickly to the right for each move, resulting in an increased bank breadth. This occurs again from $t = 3842$ sec to 3850 sec. The reason this occurs, is that the peripheral view filter to the right of center, (filter 5) is taking large steps towards a_{\max} in an effort to cover the periphery, whereas the peripheral view filter to the left of center, (filter 1) is taking small steps away from a_{\min} in an effort to maintain its position in the periphery. One solution would be to reduce the peripheral view of the bank by assigning a larger Chi-Squared probability value, P_{χ_1} , for filter 1 and/or assigning a smaller Chi-Squared probability value, P_{χ_5} , for filter 5 (see discussion in Section 3.2). The potential drawback to this idea is that a reduced peripheral view could increase the time required to converge on a large change in the true parameter value. On a positive note, for $t = 3820$ sec to 3850 sec, the parameter error mean shown in Figure 97 is seen to be relatively small (≈ -200) with a 1 sigma *magnitude* of ≈ 400 . These statistics imply that the severe overestimation and underestimation of the parameter value do *not* exist for *every* Monte Carlo run. In fact, observation of each run showed that only three of the ten runs suffered from this problem during this time frame, while the other seven runs generated very good parameter estimates. Nevertheless, it is important to identify this potential liability through the single run plot at the top of Figure 97.

Case 3 : The same conclusions drawn in cases 1 and 2 apply here, so the performance plots are not shown.

Case 4 : The PBDM selects filter-assumed parameter values which give the MMAE the ability to react to abrupt changes in the true parameter, as seen in the plot of the parameter estimate in Figure 98 from $t = 3800$ sec to 3810 sec. Similar performance is seen at $t = 3850$ sec. The erratic bank

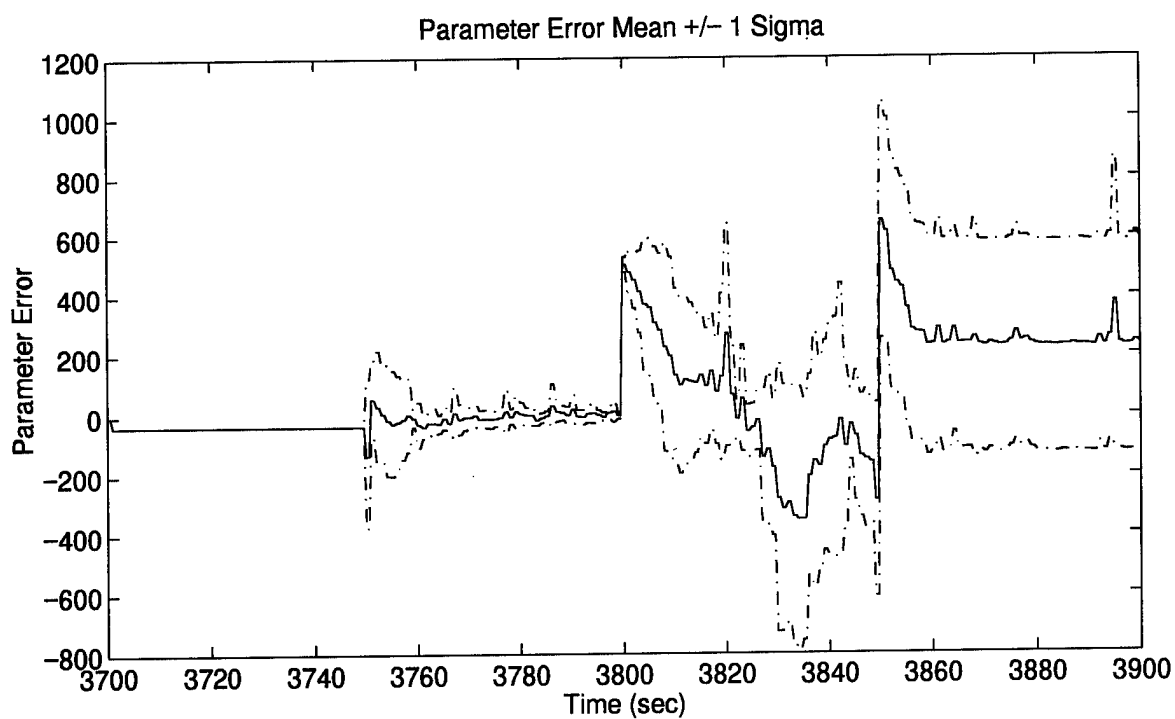
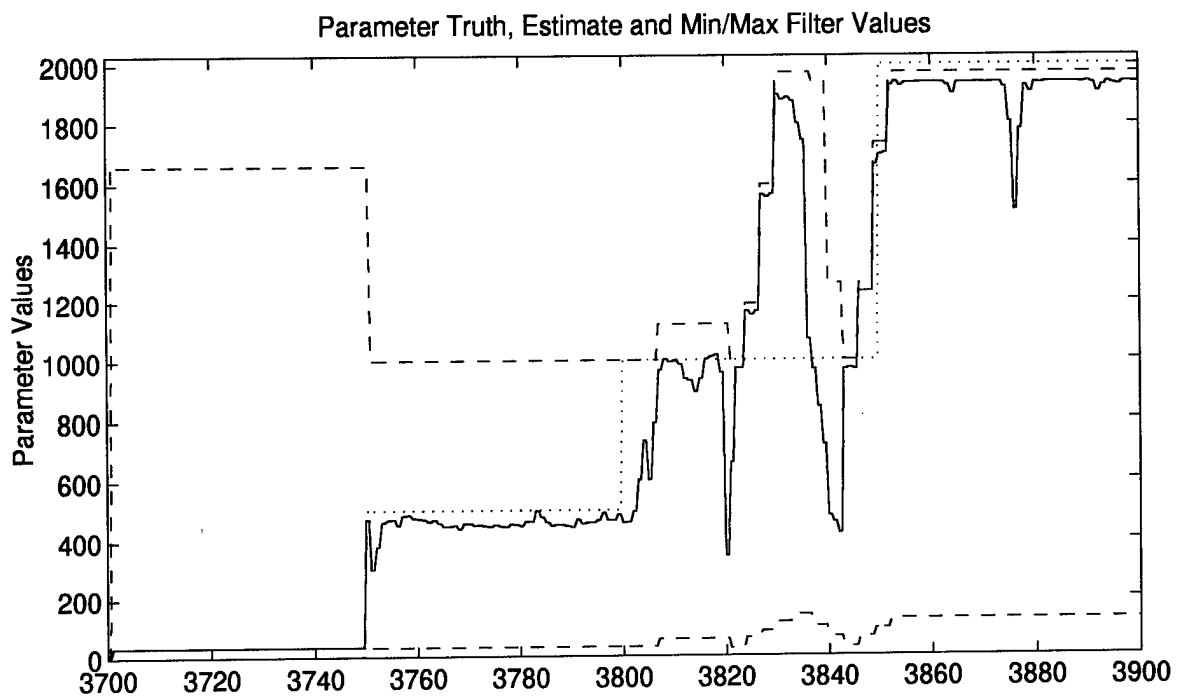


Figure 98. Parameter Estimation Performance — Case 4: Probability Algorithm

movement discussed above is still evident for $t = 3820$ sec through 3842 sec. Again, performance suffers when the bank size is too large, allowing severe overestimation of the parameter estimate ($t = 3820$ sec to 3833 sec), even though the parameter estimate is *not* at the edge of the parameter space spanned by the filter bank, i.e., $\hat{a}_{\text{MMAE}} \neq a_1$ and $\hat{a}_{\text{MMAE}} \neq a_J$. The error state estimates are shown in Figures 99 and 100 for completeness.

Case 5: The parameter estimate plot in Figure 101 further supports the claim that the PBDM provides good bounding of the parameter estimate as seen by the trace pair (---).

4.10 MMAE Incorporating the Density Algorithm with Probability Discretization

4.10.1 Implementation Issues

The final moving-bank MMAE algorithm combines the basic density algorithm with the PBDM, as discussed in Section 3.3. The design parameters used here are the same as those used for the basic density algorithm summarized in Table 6 (page 178), with one exception. The decision delay time is increased to 3 sample periods to prevent erratic changes in the filter-assumed parameter values and associated degradation of the parameter estimates. The PBDM with the probability values shown in Table 7 (page 239), maintains a broad view of the parameter space. This broad view results in a larger parameter breadth than that obtained with the stand-alone density algorithm. As a result, immediately following a moving-bank decision, the algorithm is susceptible to making an erroneous decision such as move hard left or hard right. This may occur if one of the outlying filters (the peripheral view filters) *temporarily* indicates a need to move. To counter this, the decision delay time is increased to 3 sample periods, allowing the bank to settle out transients and make better decisions.

Expansion decisions are suppressed ($T_3 = 5$), since the PBDM keeps two filters positioned with a peripheral view of the parameter space, thereby precluding the need for expansions. Note that the choices to suppress expansions and increase the decision delay are dependent on the probability

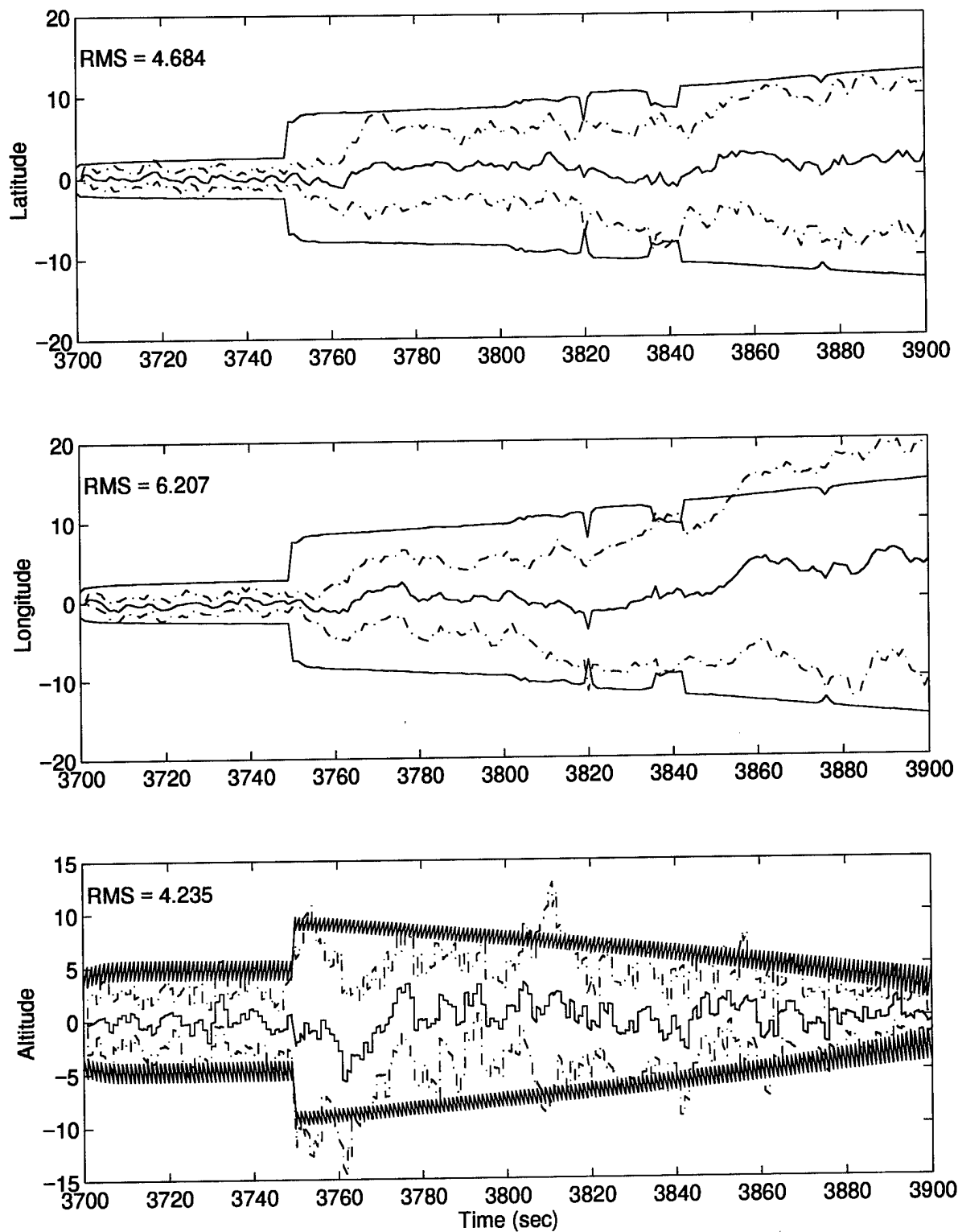


Figure 99. MMAE State Estimation Errors (feet) – Case 4: Probability Algorithm

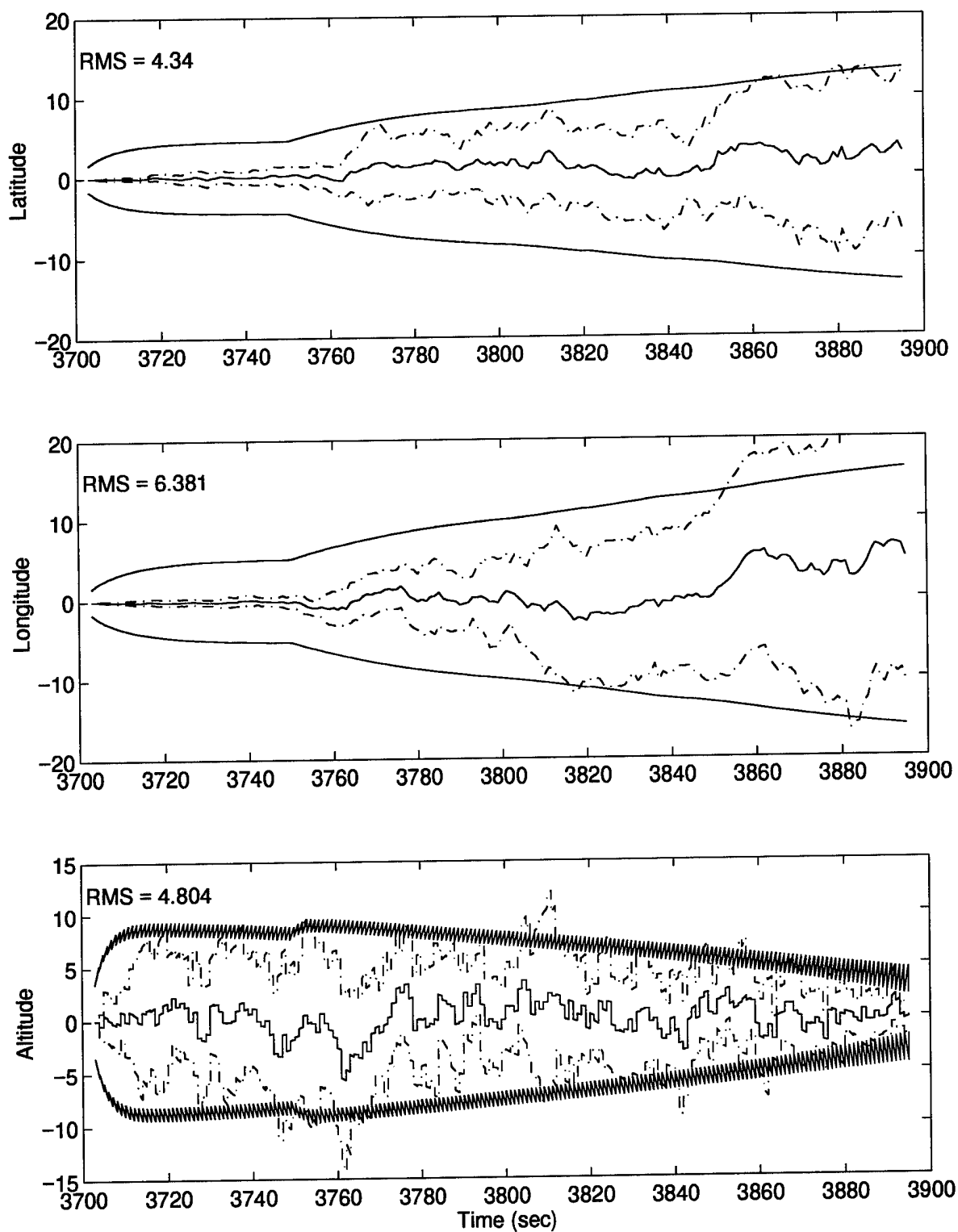


Figure 100. M³AE State Estimation Errors (feet) – Case 4: Probability Algorithm

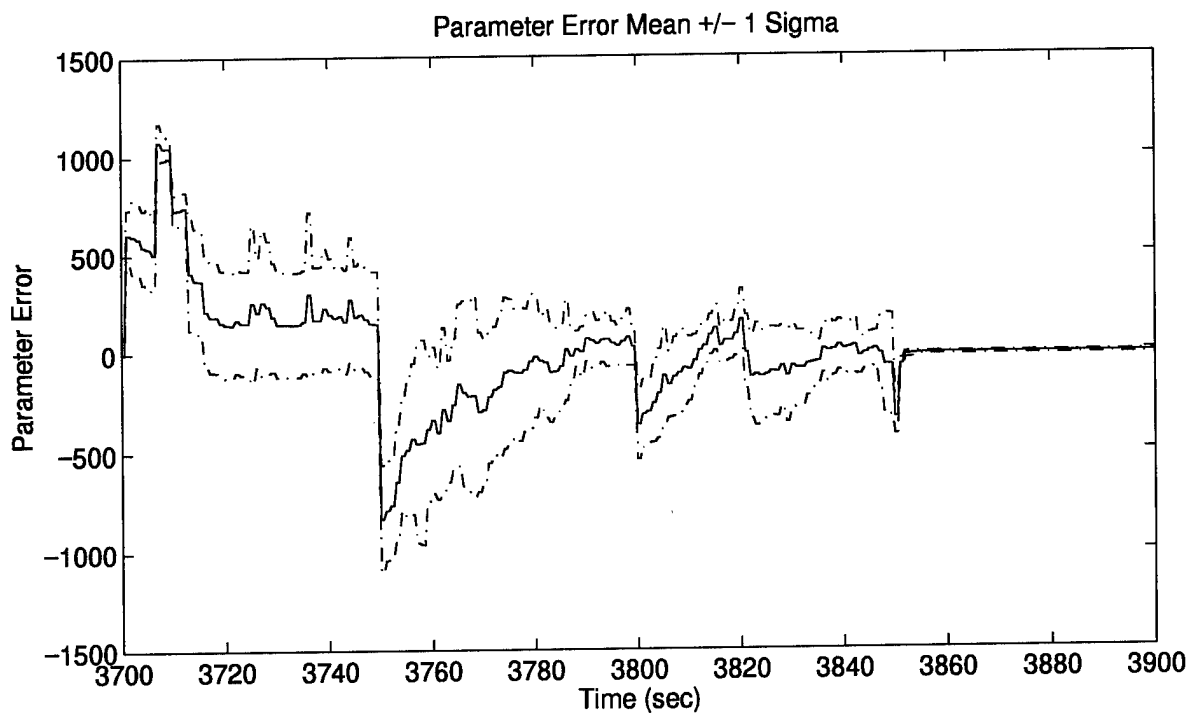
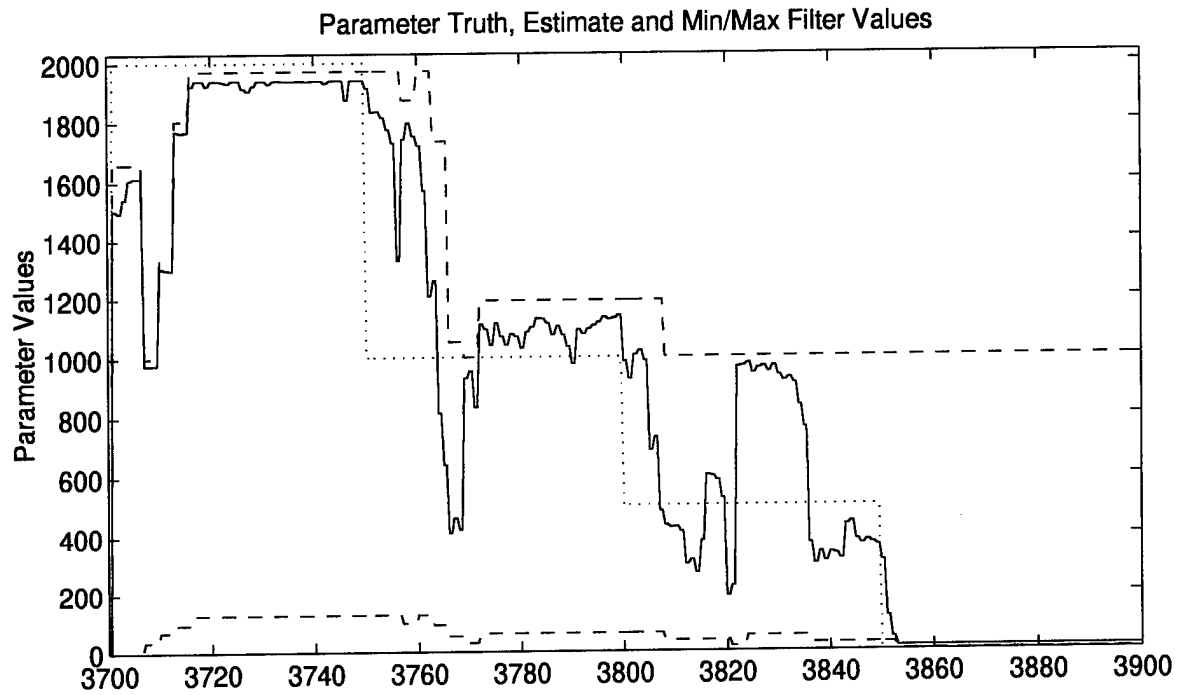


Figure 101. Parameter Estimation Performance – Case 5: Probability Algorithm

values used by the PBDM shown in Table 7 (page 239). Other probability values, resulting in a smaller parameter breadth for the bank, might affect these choices. The same 40-by-5 look-up table of parameter values presented in Section 4.9 was used on-line.

4.10.2 Performance

Case 1 : This algorithm provides the best overall performance for this test case as compared to the other algorithms being analyzed. The parameter plot in Figure 102 shows both quick convergence to the true parameter value and consistency of the estimate in the presence of a non-changing truth value. The one exception occurs at $t = 3820$ sec, which results from the unusually large (or small) noise values that have affected all the algorithms. The probability plot is shown in Figure 103 for completeness, along with the MMAE and M³AE error state estimates in Figures 104 and 105. Notice the absence of transient behavior at $t = 3850$ sec in Figure 104 as compared to Figure 95. This improvement is attributed to the quick response time of the density algorithm decision-making process. The evidence of this algorithm's superior performance is found in Tables 16 – 22 (Appendix E) of the measures \hat{e}_{RMS}^x and \hat{e}_{RMS}^a .

Case 2 : Although some of the erratic bank movement is removed as seen by comparing Figure 106 (moves based on the density algorithm) to Figure 97 (moves based on parameter position estimate monitoring), there is still room for improvement. Specifically, a series of soft moves to the right increase the breadth of the bank unnecessarily.

Case 3 : No additional insights are gained from this case study. The same conclusions drawn in cases 1 and 2 apply here, and performance plots do not warrant being shown.

Cases 4 and 5 : The trends here match those for case 4 under the MMAE incorporating the probability algorithm. The erratic bank movements previously caused by the parameter position estimate monitoring are still present, but are now the result of the density algorithm decision-making

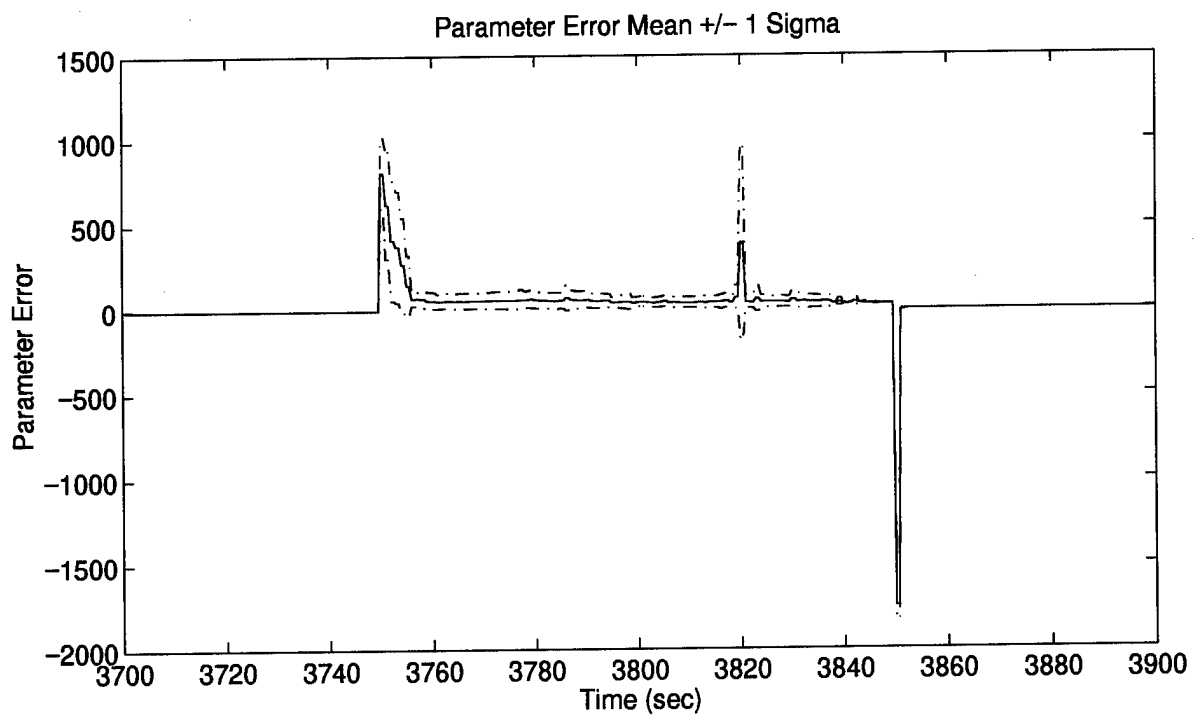
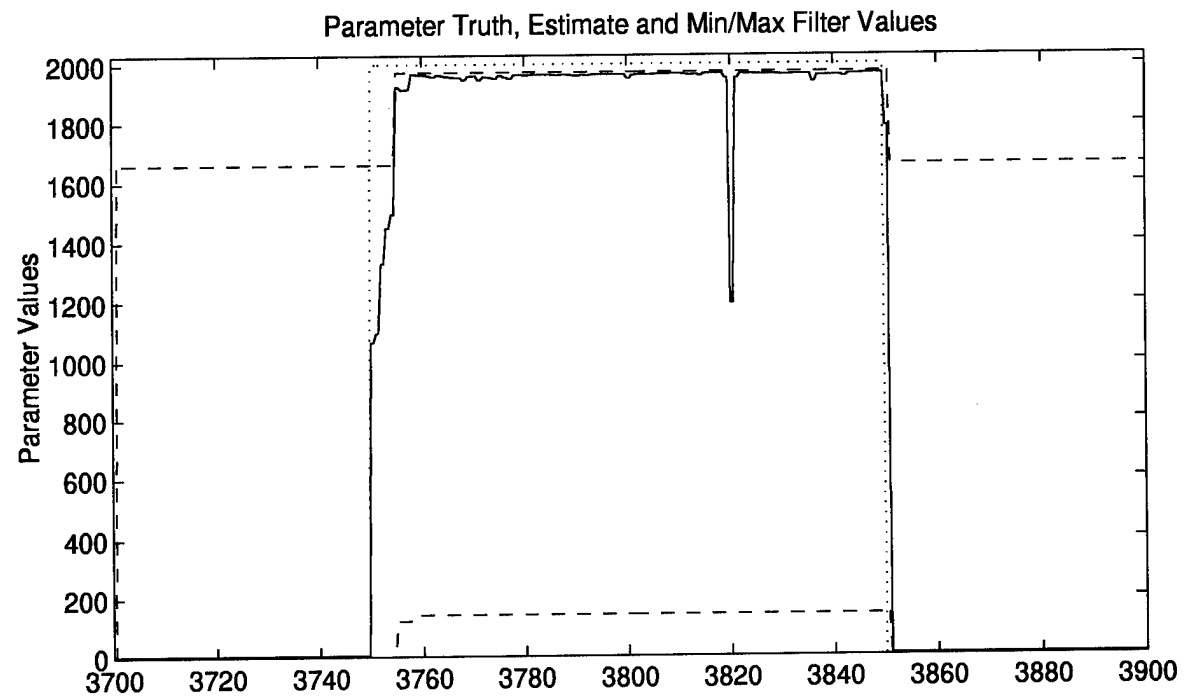


Figure 102. Parameter Estimation Performance – Case 1: Density / PBDM

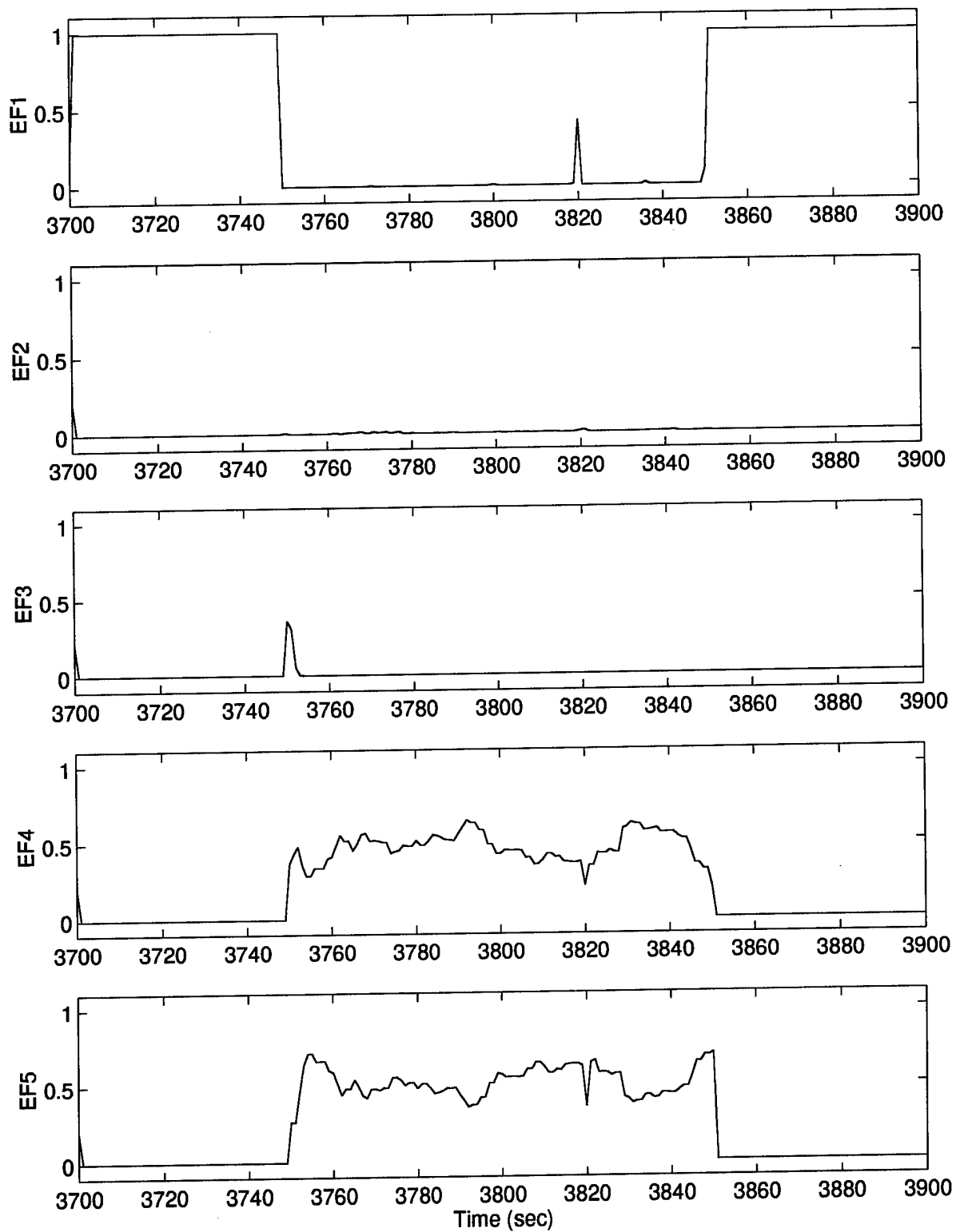


Figure 103. Elemental Filter Probabilities – Case 1: Density / PBDM

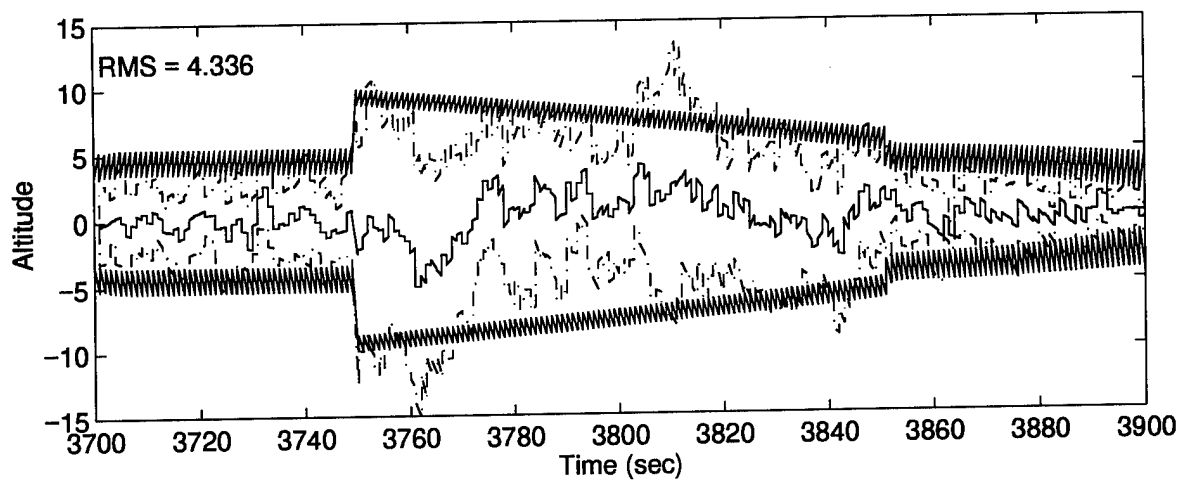
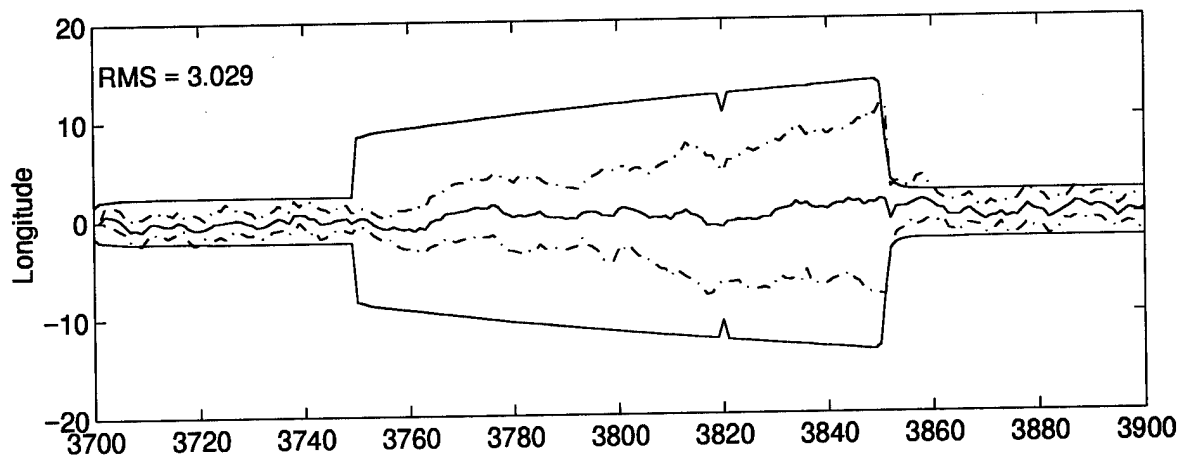
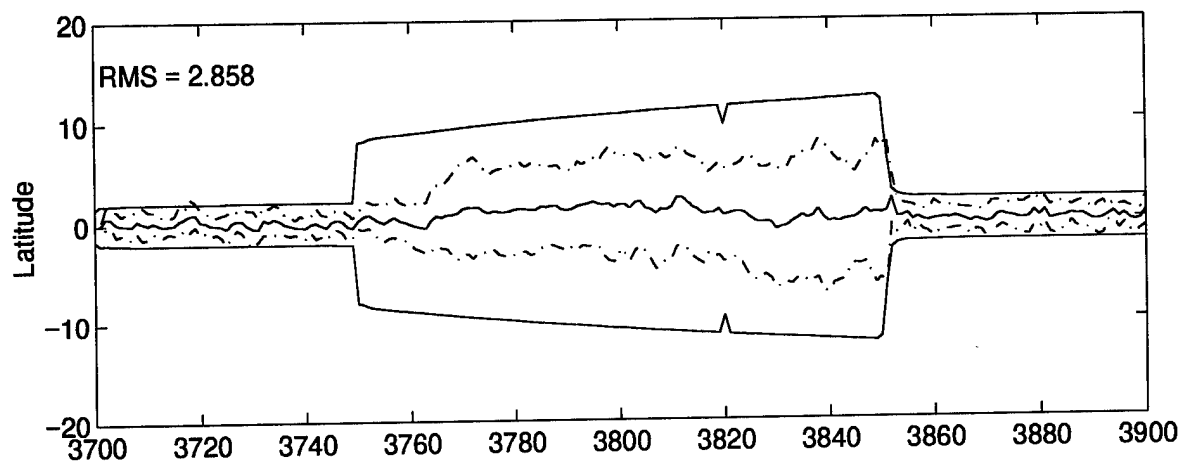


Figure 104. MMAE State Estimation Errors (feet) – Case 1: Density / PBDM

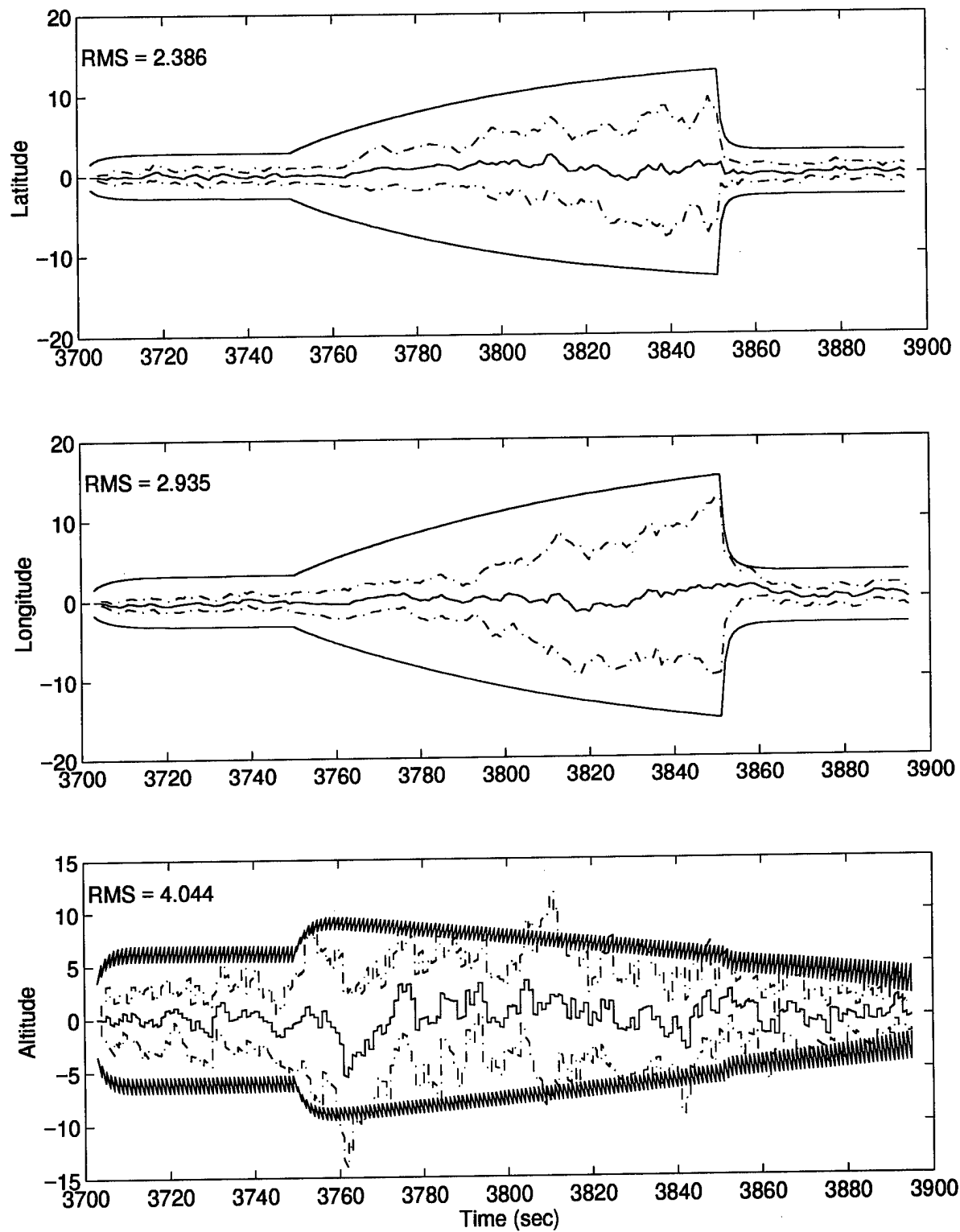


Figure 105. M³AE State Estimation Errors (feet) – Case 1: Density / PBDM

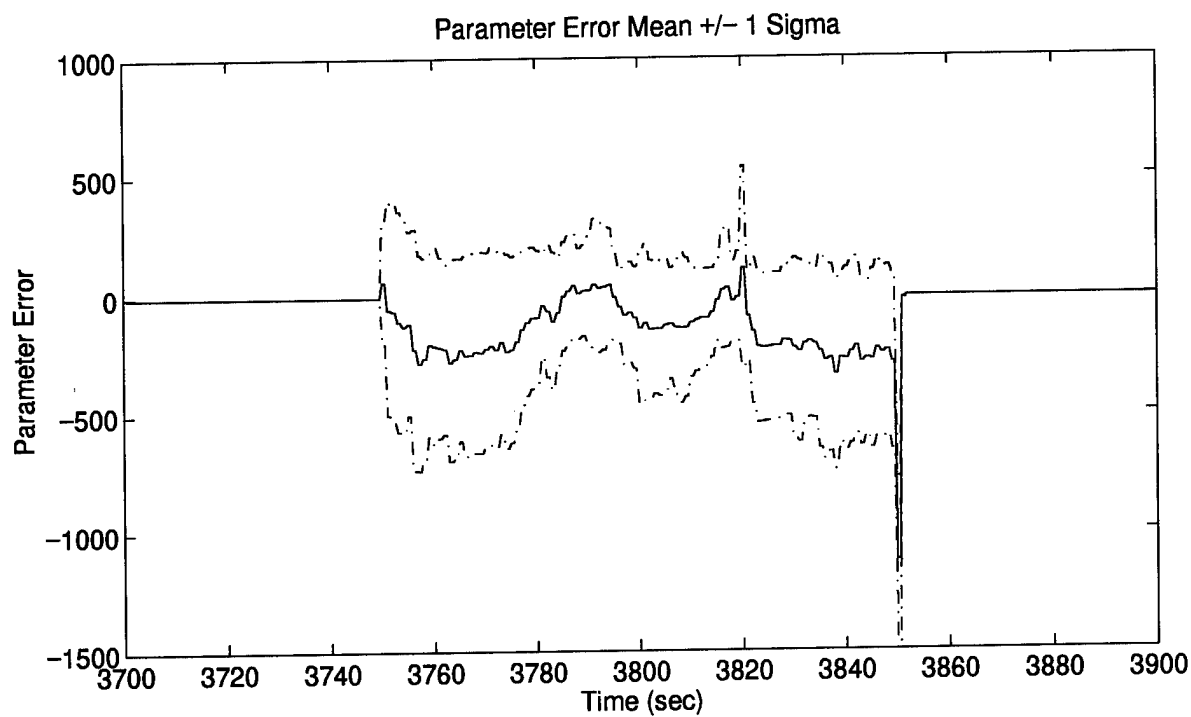
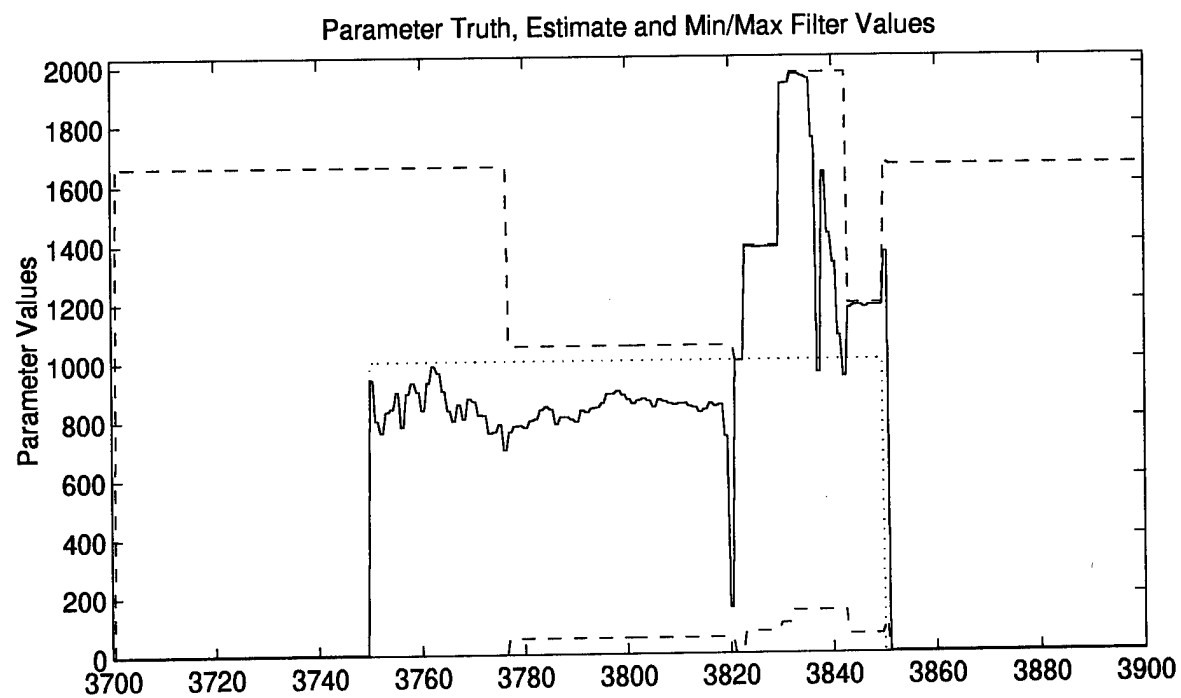


Figure 106. Parameter Estimation Performance – Case 2: Density / PBDM

process. Again, the performance plots do not provide any additional insights and are not shown for brevity.

4.11 Baseline Without Interference/Jamming

The unjammed test case (case 6) is presented as a baseline for comparison to the other interference/jamming scenarios and reveals a negative effect of lower bounding the probabilities. This case is presented in a separate section since all the algorithms produce identical results for this case. Figures 107 and 108 show the consistent bias in the parameter estimate resulting from blending all five filter-assumed parameter values when in fact, filter 1 is a perfect match to truth. Although the MMAE state estimate errors shown in Figure 109 indicate adequate tuning and selection of the filter model that best matches truth, the M^3 AE state estimate errors in Figure 110 show that the single filter in the M^3 AE architecture is too conservatively tuned. One alternative that would overcome the bias on the parameter estimate due to the probability lower bounding is to exclude the filters with a “low” probability from the blending process. In other words, the lower bounding is employed in the hypothesis conditional probability calculation to prevent filter “lock-out” as described in Section 2.2.2, but filters either at the lower bound or below some empirical threshold would be excluded from the estimate blending process described by Equations (50) and (51). Note that the probability weight associated with the filters that are excluded from the blending process is distributed proportionally among the remaining filters such that the sum of the probability of the remaining filters is equal to one. Previous research by [18, 50, 67] invoked this idea for a control problem to prevent filter lock-out while lowering the chance of applying inappropriate control input caused by the blending process.

In order to demonstrate the potential benefits gained by excluding the filters with a “low” probability from the blending process, the fixed-bank algorithm was implemented using this process.

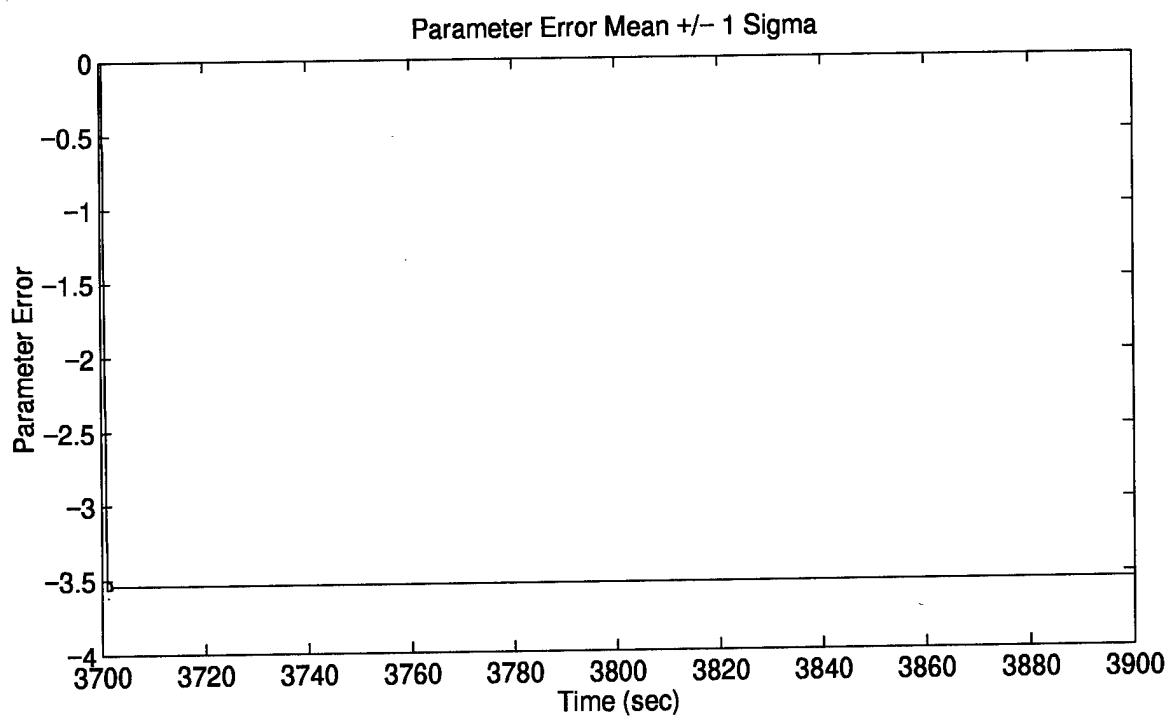
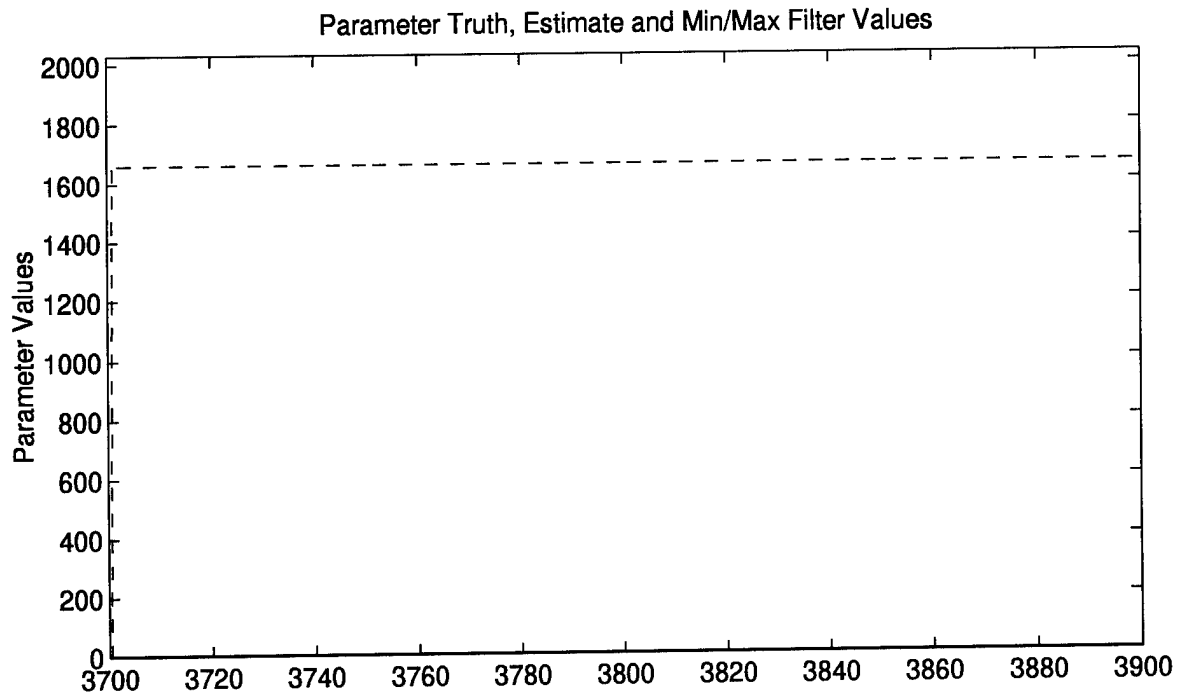


Figure 107. Parameter Estimation Performance – Case 6: No Jamming

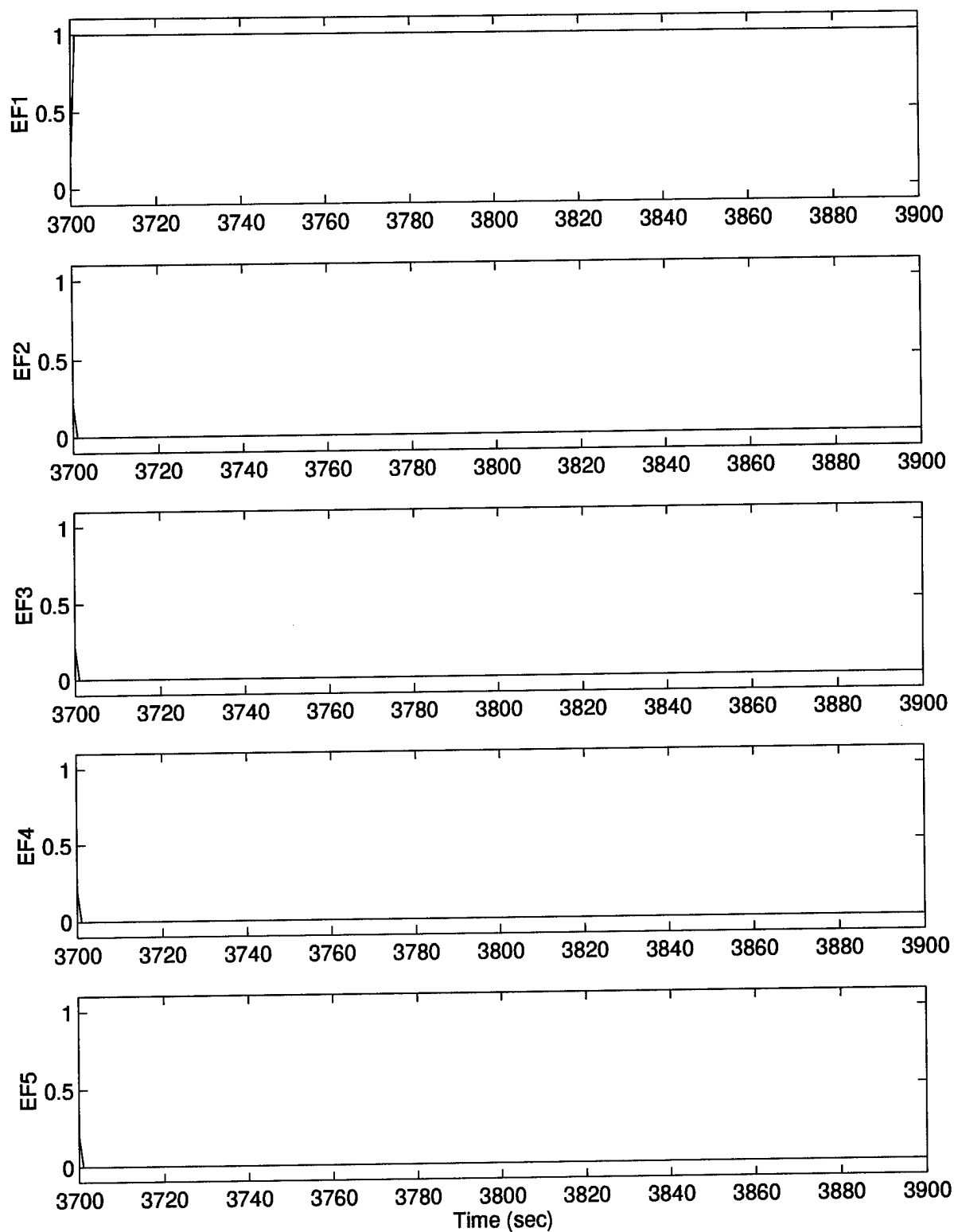


Figure 108. Elemental Filter Probabilities – Case 6: No Jamming

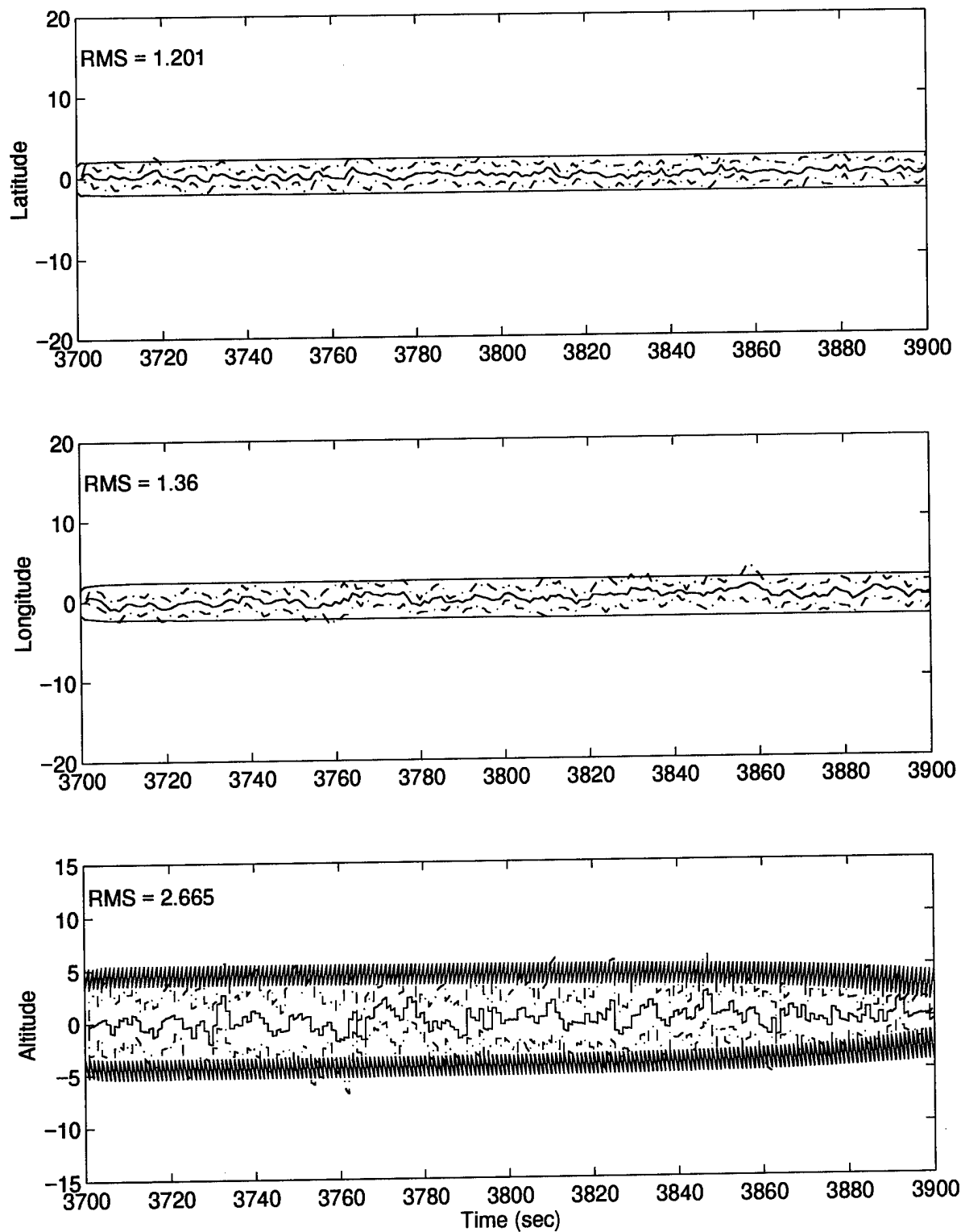


Figure 109. MMAE State Estimation Errors (feet) – Case 6: No Jamming

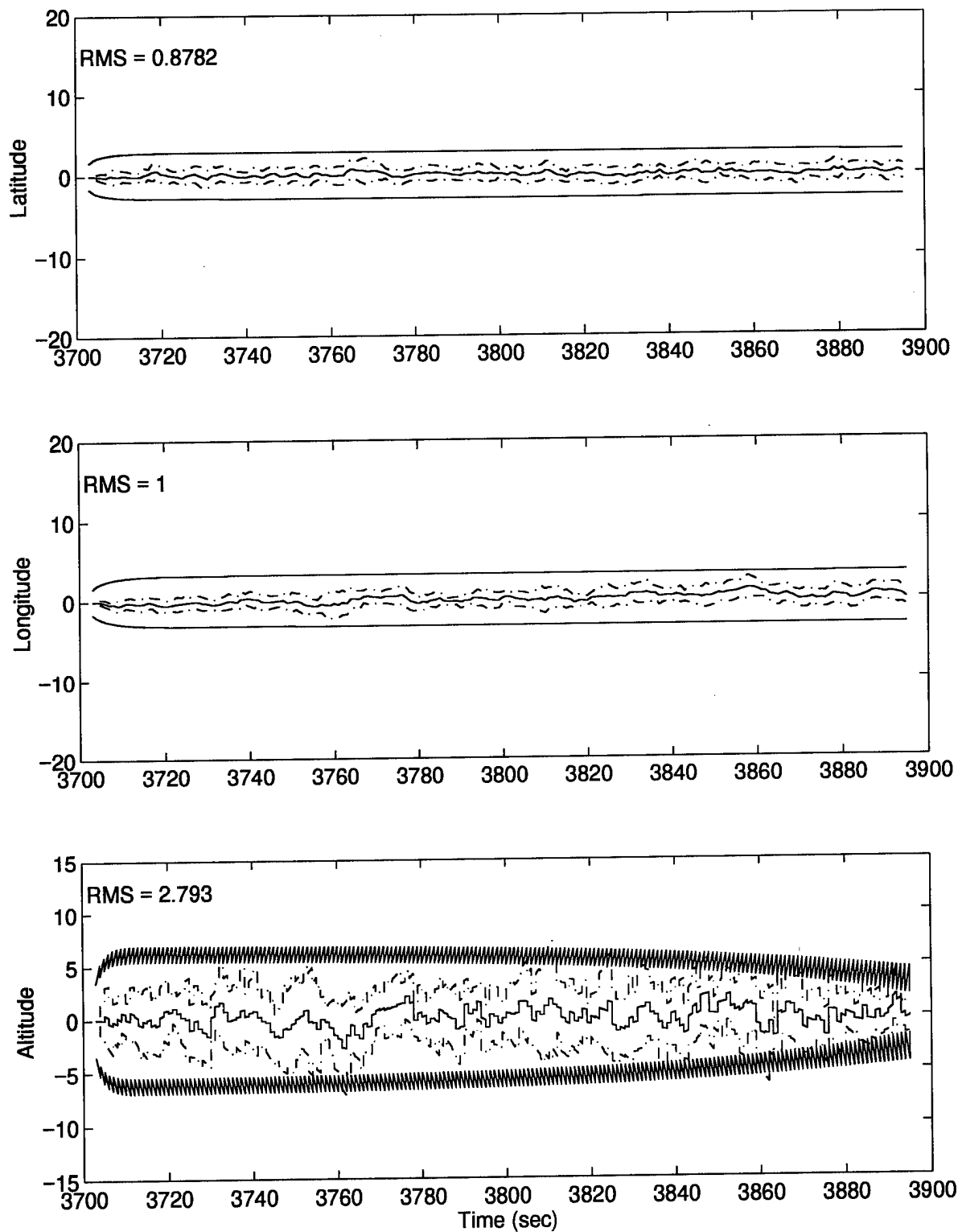


Figure 110. M³AE State Estimation Errors (feet) – Case 6: No Jamming

The algorithm choice is not important since all the algorithms perform the same under nominal conditions (no interference/jamming). The empirical threshold used to distinguish between probabilities that are considered too low to be included in the blending process was arbitrarily selected at $2 \cdot p_{\min} = 0.002$. The state estimation errors are shown in Figure 111 for the MMAE and Figure 112 for the M³AE. Note that Figure 111 is nearly identical to Figure 109, indicating the minor impact of the probability lower bounding on the MMAE state estimates. In contrast, Figure 110 shows the overly conservative tuning present in the M³AE estimates due to the lower bounding being applied to the parameter estimate being sent from the MMAE to the single Kalman filter, whereas the modified blending process (see Figure 112) does not suffer from this conservatism since the filters residing at the lower bound did not influence the blended parameter estimate. Therefore, it is highly recommended to apply the modified blending process. Furthermore, a recommendation is made in Chapter 5 to apply the modified blending approach for the other case studies in which interference/jamming exists *and* when the breadth of the MMAE bank is particularly large. For instance, algorithms implementing the PBDM designed to keep filters in the periphery of the parameter space could benefit from the modified blending process, since these peripheral filters would not skew the parameter estimate high or low, but they would still be on-line waiting for an abrupt change in the true parameter.

Finally, comparison of Figure 111 (MMAE state estimates) to Figure 112 (M³AE state estimates) reiterates the observations made on page 198, that the M³AE will significantly outperform the MMAE *only* when there is significant *blending* of more than one elemental filter. A quick look at the probability plot associated with the modified blending (Figure 113) shows that significant blending is not present and explains the similarity in the MMAE and M³AE state estimates.

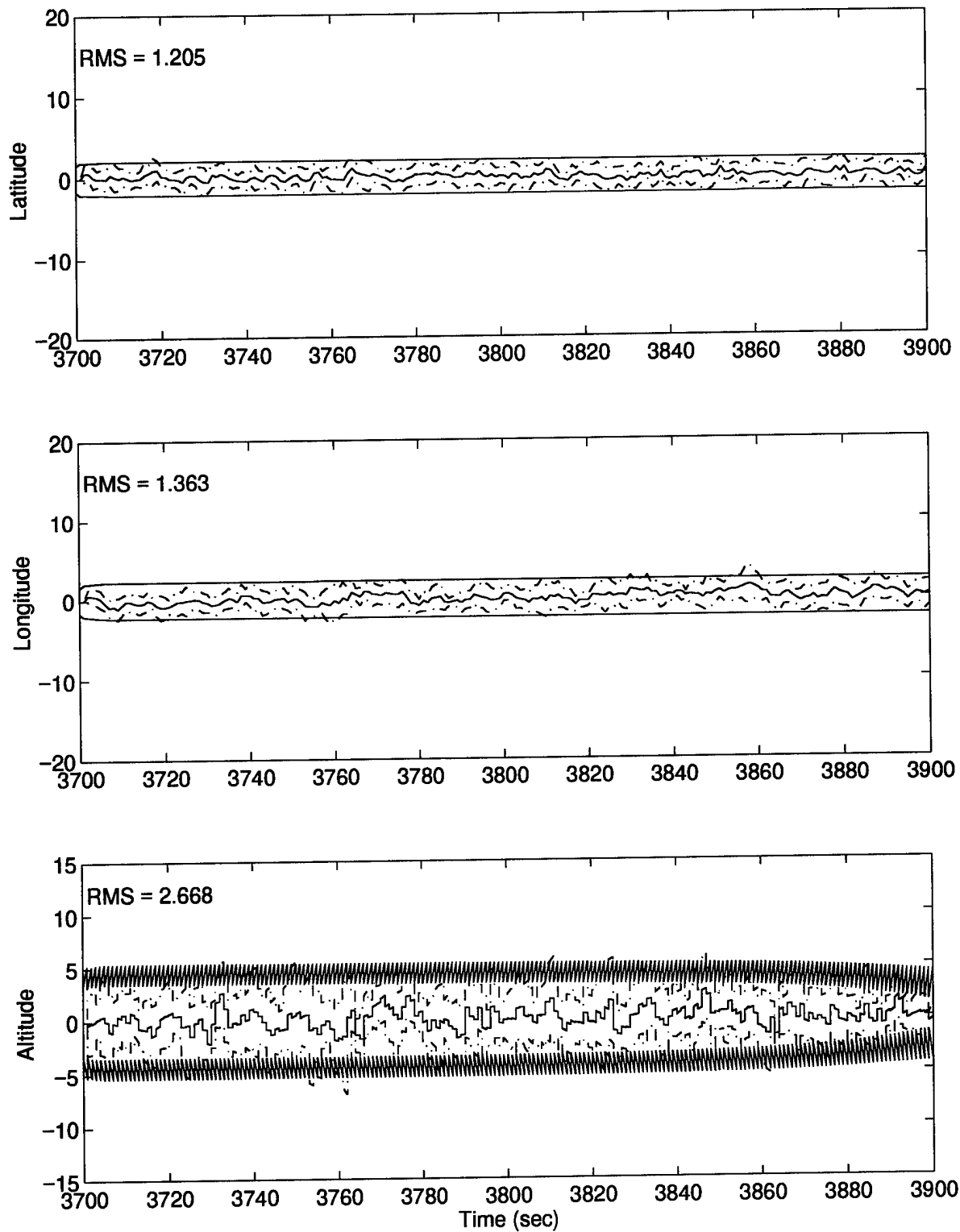


Figure 111. MMAE State Estimation Errors (feet) — Case 6: No Jamming and Modified Blending

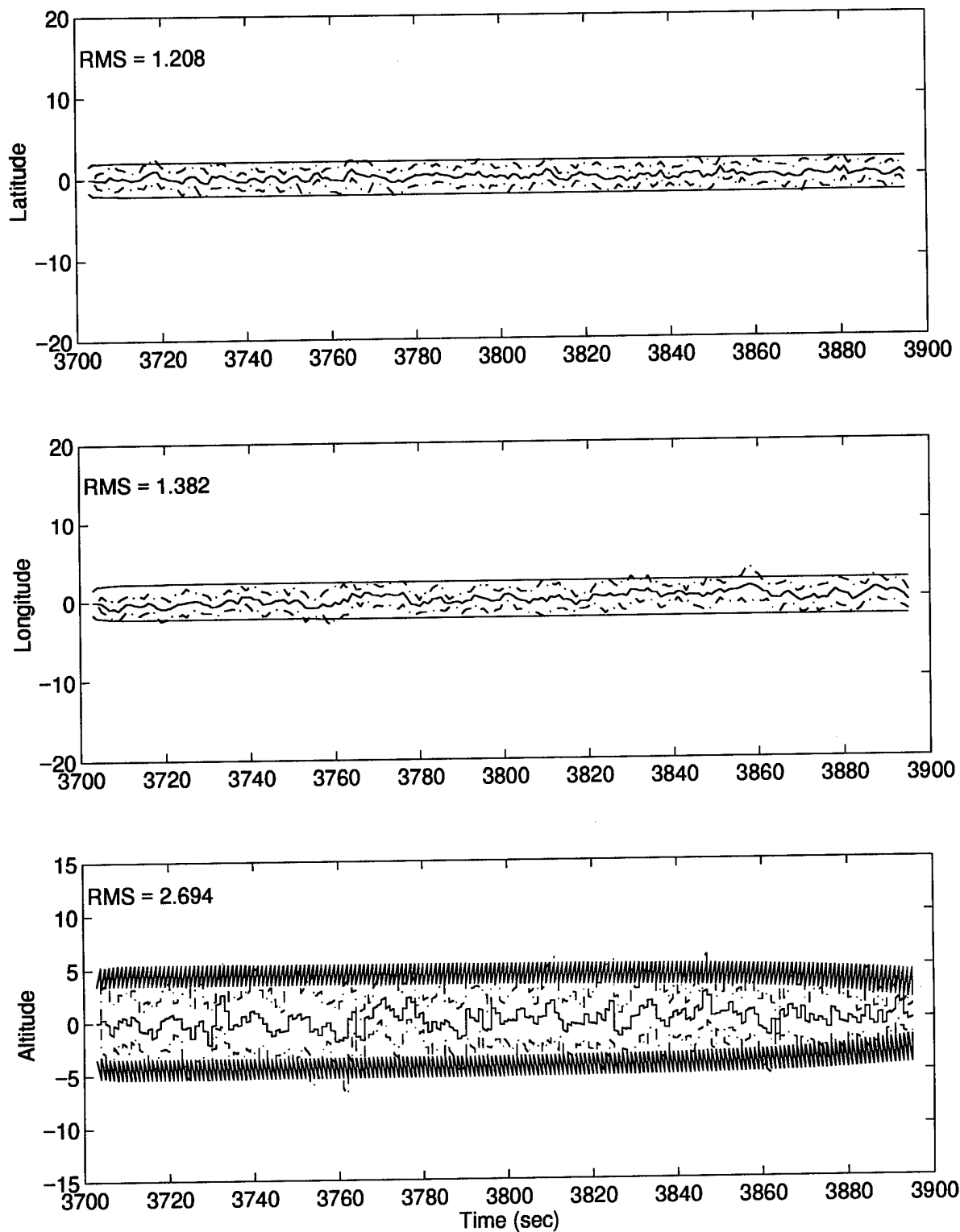


Figure 112. M³AE State Estimation Errors (feet) – Case 6: No Jamming and Modified Blending

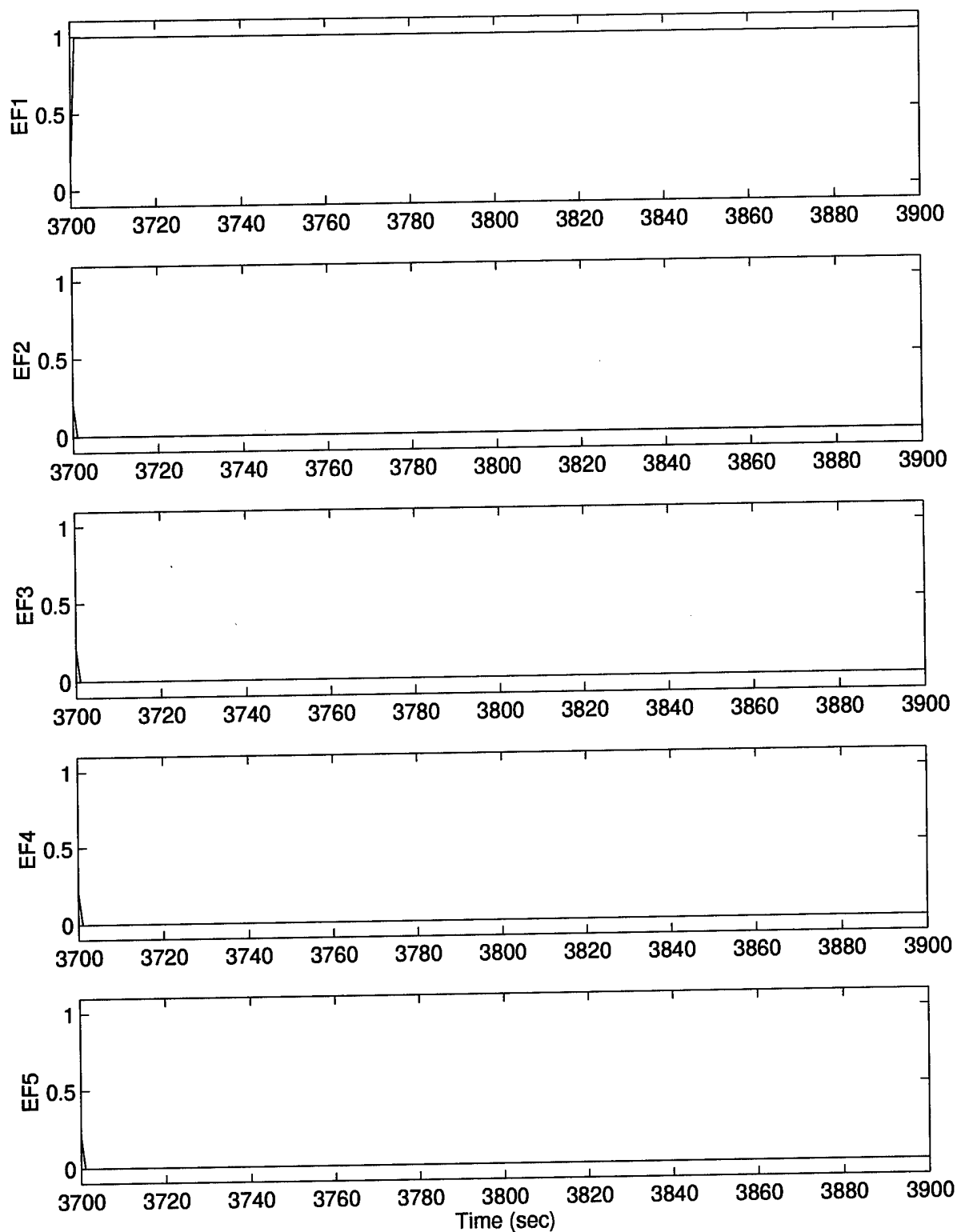


Figure 113. Elemental Filter Probabilities – Case 6: No Jamming and Modified Blending

4.12 Single Kalman Filter

A single Kalman filter with $R_{GPS} = R_0$ (i.e., no interference/jamming assumed) was implemented for comparison purposes. The inability of this single filter to adapt to true parameter changes makes it victim to poor state estimation in the presence of interference/jamming. Although a seemingly unfair comparison, the degraded state estimation performance in the presence of interference/jamming motivates the need for the adaptive algorithms being presented here. Error state estimation plots for cases 1 and 5 are shown in Figures 114 and 115 as representations of the degraded performance encountered with the use of a single Kalman filter without the benefit of being provided accurate parameter estimates. Note the dramatically different scales on these plots, compared to all previous state estimation error plots. The measure, \hat{e}_{RMS}^x , is printed on each plot and further indicates the poor performance.

4.13 Chapter Summary

This chapter presented the models used for the example problem simulated in software. The various design parameters for the algorithms were identified, along with numerous implementation issues relative to each algorithm. Simulated data based on the aircraft precision landing application was plotted and tabulated, to demonstrate the enhanced performance achievable through the new moving-bank MMAE methodologies. Although some comparison of the algorithms was accomplished in this chapter, final conclusions are made in Chapter 5 along with recommendations for future research.

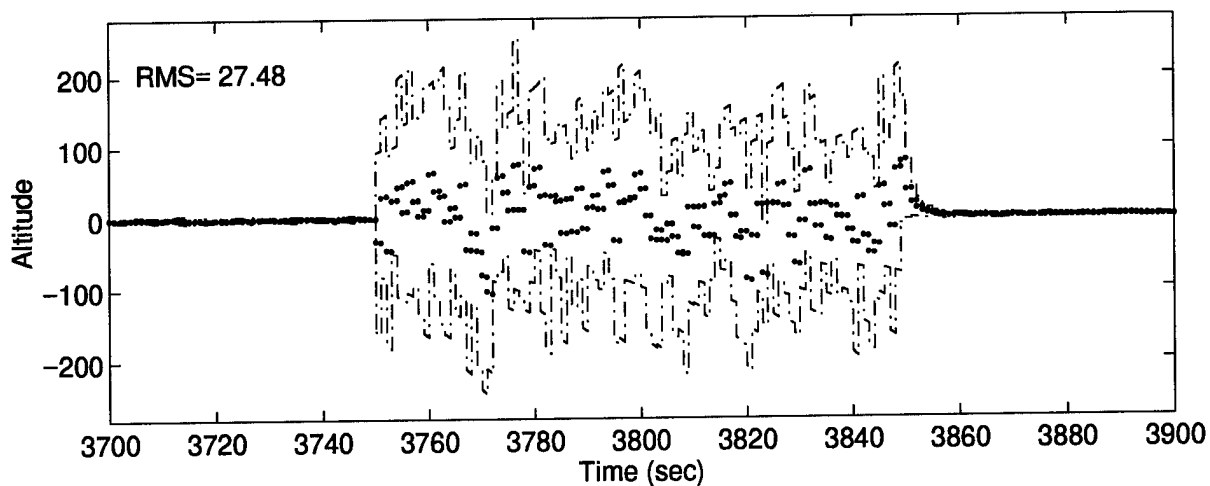
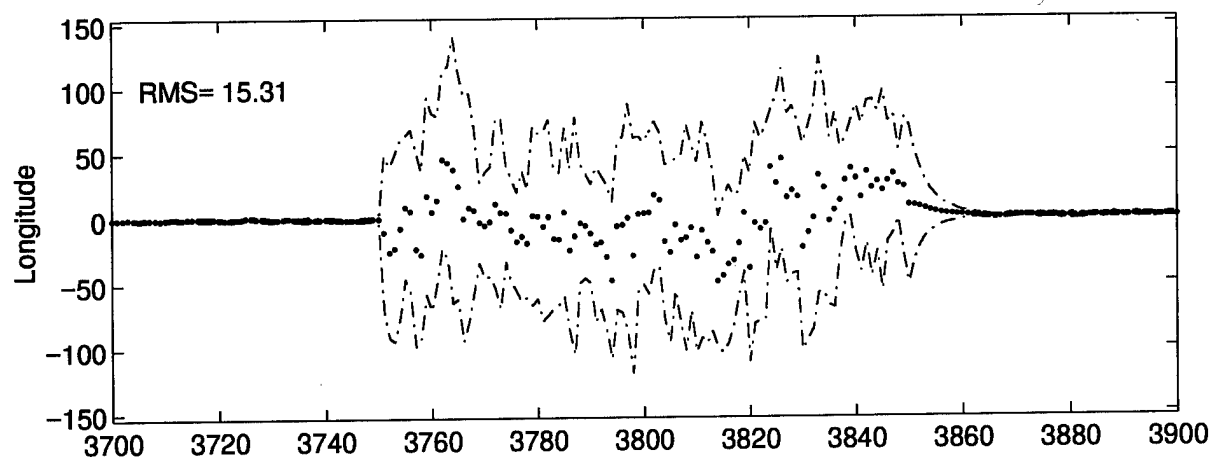
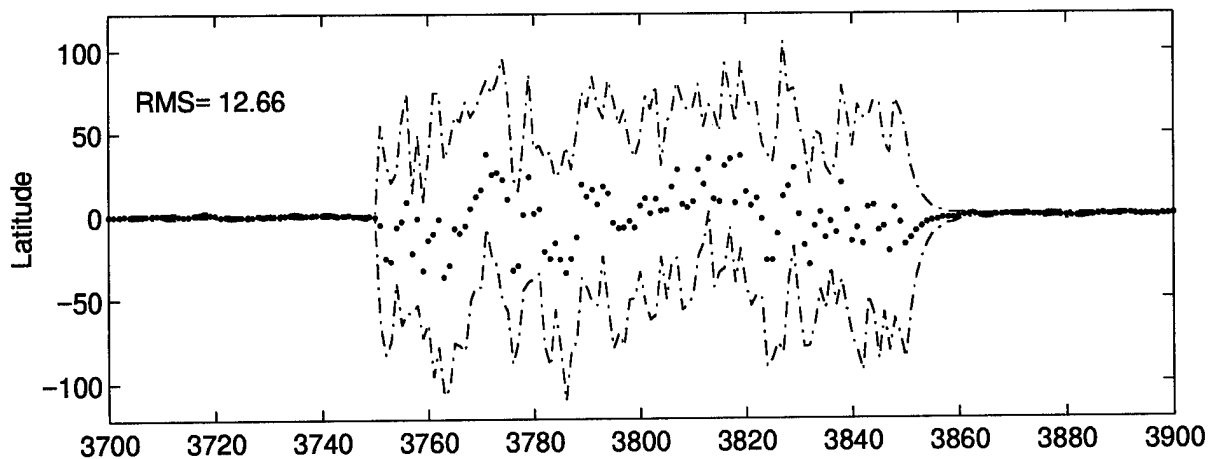


Figure 114. State Estimation Errors (feet) – Case 1: Single Kalman Filter

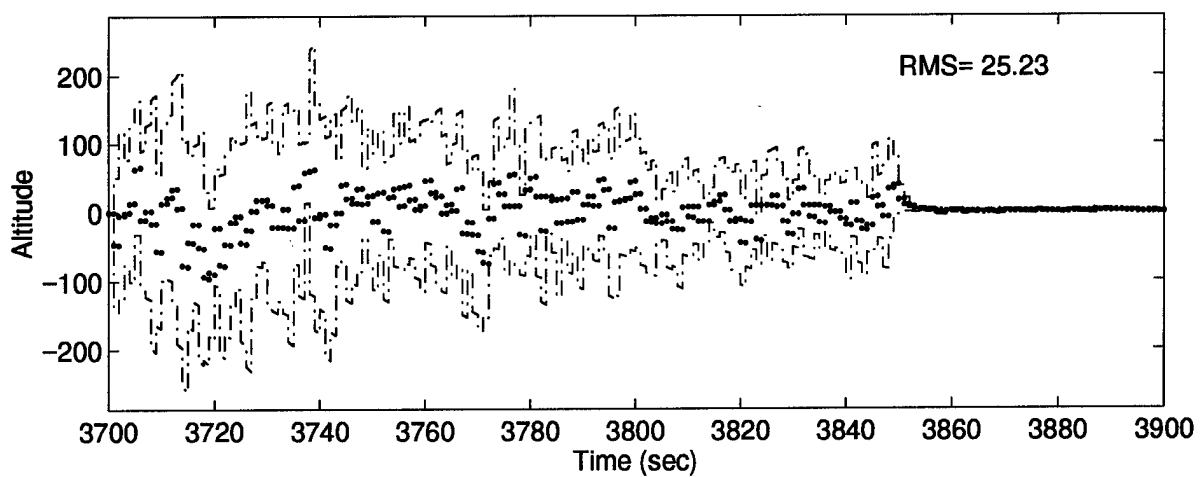
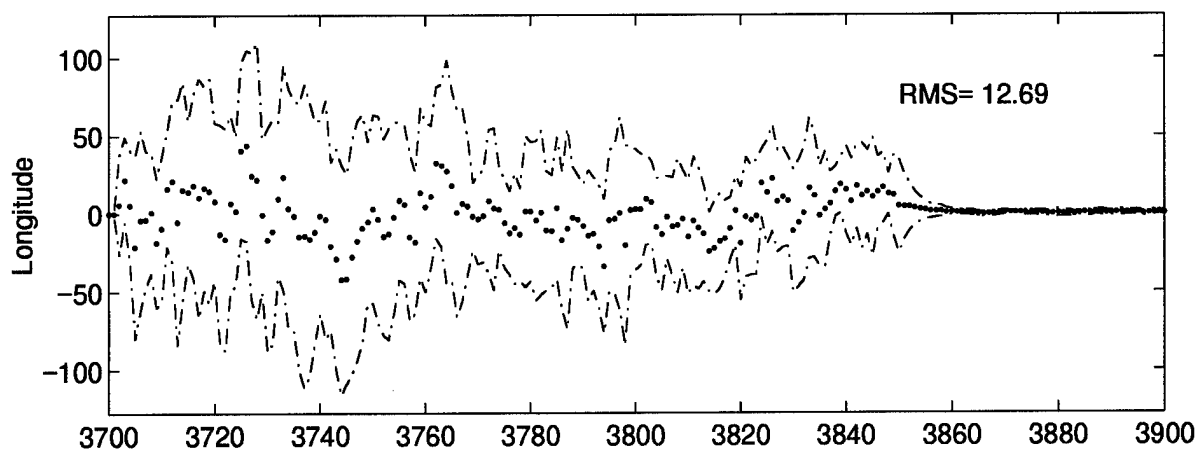
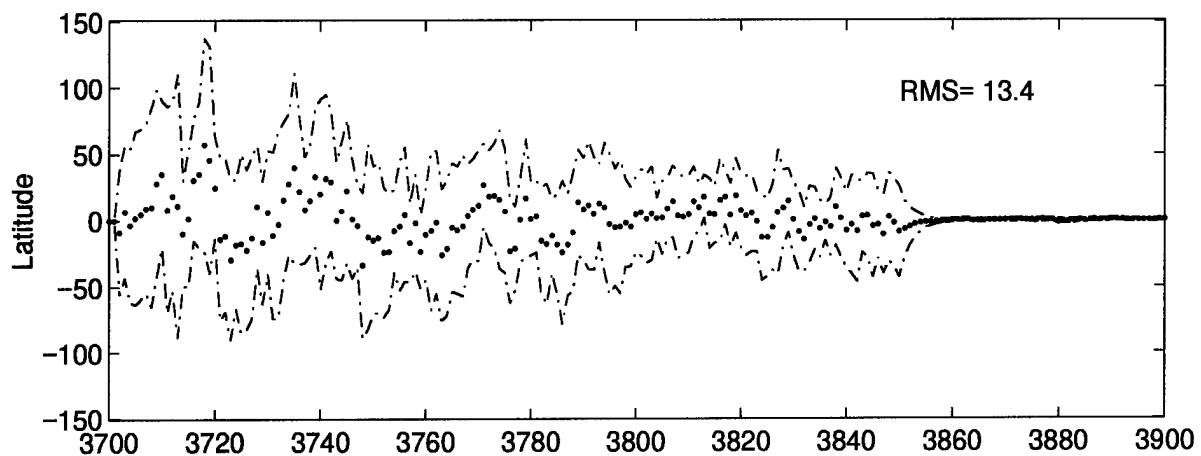


Figure 115. State Estimation Errors (feet) – Case 5: Single Kalman Filter

Chapter 5 - Conclusions and Recommendations

5.1 Conclusions

Prior to this research, limited methods were available to determine moving-bank decisions or parameter discretization adequately within the structure of multiple model adaptive estimation (MMAE). Most of these methods rely on logic which utilize *ad hoc* thresholds determined through empirical analysis. The goal of this research is to replace these existing *ad hoc* techniques with analytically-based algorithms, and the contributions made by this research provide designers with new flexibility when selecting bank moving or sizing methods for a moving-bank MMAE. In particular, this research has led to the development of new algorithms and modifications to existing algorithms associated with moving-bank MMAE. These new algorithms exploit information not previously utilized in a moving-bank MMAE and provide an analytical basis for some of the thresholds used in the decision-making logic. Additionally, a new parameter discretization method was developed to be used in conjunction with a moving-bank MMAE algorithm.

Chapter 4 demonstrated one application of these algorithms to an aircraft GPS-aided INS navigation system subjected to interference/jamming while attempting a successful precision landing of the aircraft. In some respects, this example problem is not as interesting as some others, in that state estimation performance did not always vary significantly from one algorithm to another. This is primarily due to the relatively benign flight environment that exists for the landing approach portion of the flight profile being simulated *and* the choice of the unknown parameter. Specifically, the GPS measurement noise variance is chosen as the unknown parameter, and once a significant level of interference is induced (i.e., $R_{GPS} \geq 500$), the GPS signal is effectively lost along with the feedforward corrections provided from the GPS to the INS. For example, assuming $R_{GPS} = 750$ when in reality, $R_{GPS} = 500$, the associated Kalman filter is too conservatively tuned, but the state

estimation performance does not suffer appreciably. Therefore, poor parameter estimation does not always result in poor state estimation, even with the recently developed M³AE architecture which focuses on enhancing state estimation through enhanced parameter estimation. The state estimate performance measure, \hat{e}_{RMS}^x (the temporally averaged RMS value of the state estimation error), listed in Tables 17 – 22 of Appendix E, further illustrates this point through both a lack of variation in the measure values and an inconsistency between superior parameter estimation and superior state estimation. Despite the lack of variation in the state estimation performance, the parameter estimation performance is much more distinct between the algorithms, as illustrated by the measure \hat{e}_{RMS}^a (the temporally averaged RMS value of the parameter estimation error) in Table 16. Therefore, final conclusions are focused on the parameter estimation measure.

Recall case 2, in which the true GPS noise interference level of $R_{GPS} = 9000 ft^2$ (versus the nominal noise level of $R_{GPS} = 9 ft^2$) was induced for 100 sec of the 200 sec flight profile. For this case, the fixed-bank MMAE outperforms all of the moving-bank algorithms. However, for all other cases, it is outperformed by at least one of the moving-bank algorithms, and in most cases it generates one of the worst performance measure values. Recall that case 2 presented the scenario in which one of the fixed-bank filters was closely matched to truth. The fixed-bank performance quickly degrades when this is not the case, and given a relatively broad parameter space combined with a relatively few number of filters, it is unrealistic to assume that one of the fixed-bank filters would always lie close to truth. Therefore, the moving-bank algorithms are proven worthy of the additional computations needed for their implementation. The remainder of this discussion will focus on the moving-bank algorithms.

The basic density algorithm developed in Section 3.1 and the density algorithm with expansions and additional delay (see Section 4.7.3) consistently provide good parameter estimates that compete very well with all the other algorithms in every case study. In particular, notice that the

density algorithm with expansions and additional delay has the best value of \hat{e}_{RMS}^a for case 5 (the true noise interference/jamming level begins at $R_{GPS} = 18000 ft^2$ and undergoes three successive decreases to $R_{GPS} = 9000 ft^2$, $4500 ft^2$, and $9 ft^2$) and one of the best values of \hat{e}_{RMS}^a for all other cases. In contrast, the other moving-bank algorithms suffer large estimation errors in at least one case study while performing very well in one or two other cases. When considering whether or not to include expansions within the density algorithm's decision-making process, the main issue is the possibility of erratic bank movement that coincides with allowing expansions. This erratic movement is identified in the performance evaluation discussed in Chapter 4, as causing relatively *small* errors in the parameter estimates. Methods to reduce this erratic bank movement are discussed in Section 4.8.3. As mentioned above, given that a significant level of interference/jamming is induced, this problem's lack of sensitivity to *small* errors in the parameter estimate prevented serious degradation of the MMAE and M³AE state estimates. A different application may be more sensitive to these small parameter estimate errors, requiring the expansion decision to be modified to eliminate any and all erratic behavior, or to be removed from the algorithm altogether.

The main contribution associated with the density algorithm is the exploitation of new information (the conditional density, $f_{z_i|a, Z_{i-1}}$, of the current measurements, z_i , provided to the MMAE, conditioned on the assumed value of the parameter vector, a , and the observed values of the previous measurement history, Z_{i-1}) not previously utilized in moving-bank multiple model adaptive estimation. The methods used to exploit this information include the measures $M_1 - M_9$ of Section 3.1.3 and the decision-making logic presented in Section 3.1.4. Future researchers may need to consider other methods (such as those discussed in Section 5.2) in order to build on these first efforts.

A modification to the existing algorithm developed by Sheldon [68, 69] was presented as a joint effort with Miller [53]. Sheldon's algorithm is modified to provide on-line discretization of an adaptive MMAE bank rather than a single off-line discretization for a non-moving-bank MMAE

algorithm. Also, a previous assumption requiring steady state, constant-gain filters is replaced with a less restrictive finite horizon assumption for an iterative equation used to generate approximate or “pseudo”-constant gain values.

The modified Sheldon algorithm is used in conjunction with the density algorithm of Section 3.1, and the performance varies significantly from case to case, as indicated by \hat{e}_{RMS}^a . Expansions are much more necessary when using the on-line Sheldon discretization process, but their use is still problem-dependent. In general, the on-line Sheldon approach suffered from two implementation decisions. First, unlike all the other moving-bank algorithms, the MMAE bank is *not* centered on \hat{a}_{MMAE} since the Sheldon algorithm does not, in general, select a filter-assumed parameter value equal to \hat{a}_{MMAE} . Second, the Sheldon algorithm does not, in general, select a filter-assumed parameter value equal to a_{max} , resulting in large estimation errors when the true parameter value, a_t , is close to a_{max} . Section 4.8.3 identifies one method to rectify this problem of large estimation errors, and Section 5.2 will discuss these modifications more fully. However, these modifications will require giving “good engineering judgment” precedence over the cost minimization approach which is the basis of Sheldon’s algorithm. *Overall*, the *current* implementations using the on-line Sheldon discretization did not perform as well as the other algorithms for this example problem. The contributions include extension of Sheldon’s off-line algorithm for on-line use and application of Sheldon’s algorithm to systems with nonlinear models or linear/linearized system models that are astable or unstable.

The probability algorithm of Section 3.2 provided a stand-alone algorithm which combined parameter estimate position monitoring with the newly-developed probability based discretization method (PBDM), also discussed in Section 3.2. The fundamental concept employed by the PBDM is to choose the parameter values for each of the $j = 1 \dots J$ elemental Kalman filters in an MMAE, based on the calculation of the probability $P_x(\chi_j \leq T)$. This probability calculation is the prob-

ability that the *generalized* Chi-Squared variable, χ_j , will lie below a threshold, T , where χ_j is defined by the following quadratic form of the measurement residuals, \mathbf{r}_j , (assuming that the real world parameter value is \mathbf{a}_t) and the associated filter-computed covariance matrix, \mathbf{A}_j , based on the assumption within the j^{th} elemental filter that the parameter value is \mathbf{a}_j :

$$\chi_j = \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$$

The probability algorithm is somewhat competitive with the other algorithms in terms of \hat{e}_{RMS}^a , as is the combination of the density algorithm with the PBDM. The real contribution comes from the development of the PBDM, which is much less *ad hoc* than other discretization methods such as uniform or logarithmic parameter spacing. Additionally, the PBDM provides an attractive alternative to the Sheldon discretization method.

Additional contributions include:

1. Derivation of the first two moments of the *generalized* Chi-Squared density function, $f_\chi(c)$, in the presence of system mismodeling (see Appendix A). This contribution provides the first steps in the derivation of a general equation for this density function. Given this density function, one could replace many of the *ad hoc* techniques used for threshold selection in the density algorithm (see Section 3.1.3).
2. Development of a new numerical integration method denoted the Modified Simpson's Rules (see Appendix B). The modified Simpson's rules are a more general form of the basic Simpson's rules, allowing the spacing between the discrete samples used in the integration to be non-uniform.
3. Extending Hanlon's [22] derivations of equations for the mean of the residual in the presence of mismodeled measurement or dynamics noise covariances (see Appendix C). Clearly, mismodeled measurement or dynamics noise covariances can be motivated by many real-world

applications (including the GPS interference/jamming application used in this research), and the Kalman filter residuals provide useful information for adaptive parameter and state estimation. Therefore, characterizing the first two moments of the residuals is extremely worthwhile, as shown in the development of the PBDM of Section 3.2.

5.2 Recommendations

The discussions in Chapter 3 and Chapter 4 identified several recommendations for future research. Each recommendation will be briefly stated along with the page number where it was first proposed in order to assist the reader with the context of the recommendation.

The density algorithm utilized the likelihood quotient $\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$ and the density data $f_{\mathbf{z}|\mathbf{a},\mathbf{z}}(j)$ for each filter to generate the measures $M_5 - M_9$ given by Equations (86) – (97). An alternative use of the binary information obtained in forming these measures was first introduced on page 74 relating to measure M_7 and is repeated here. Notice that $\xi(j)$ in Equation (91) for measure M_7 associates a binary value with each elemental filter, and the concatenation of these binary values could be viewed as a binary word. Consider the three cases $[1\ 0\ 0\ 0\ 1]$, $[1\ 1\ 0\ 0\ 0]$, and $[0\ 0\ 0\ 1\ 1]$ which all result in $M_7 = 2$, but depict significantly different scenarios requiring different moving-bank decisions. In this context, the value of M_7 is less important than the location of the 1's and 0's or the decimal value associated with each binary word. For example, large binary words such as $[1\ 1\ 0\ 0\ 0]_2 = 24_{10}$ would indicate a need to move right, away from the “bad” filters, versus small binary words such as $[0\ 0\ 0\ 1\ 1]_2 = 3_{10}$ used to indicate a need to move left. This binary information could lead to decisions regarding the harshness of a bank move, as was done with measure M_7 . Furthermore, the binary information could give insights into which direction to move the bank and/or when to expand and contract the bank. This same type of binary word construct could be used to enhance the information content of any measures expressed as sums, i.e., M_5 through M_9 .

Derivation of the first two moments of the *generalized* Chi-Squared density function, $f_{\chi}(c)$, in the presence of system mismodeling was presented in Appendix A. However, the discussion on page 77 identified the benefit of deriving a general equation for the density function and applying a likelihood ratio test based on false and missed alarm rates specified by the designer. Attempts were made by this author to define the density function fully, including the use of relationships between characteristic functions and their associated density functions. However, this did not lead to a characteristic function that is directly related to any known density function(s). Therefore, continued research is needed in the development of the *generalized* Chi-Squared density function, $f_{\chi}(c)$.

The tendency of the MMAE to favor more conservatively tuned filters resulted in a biasing high of the parameter estimates, $\hat{\mathbf{a}}_{\text{MMAE}}$. A recommendation was made on page 105 to account for this bias when appropriate. One approach would be to focus on the quadratic $\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$ which affects the density function in Equation (47) and ultimately affects the probability calculation given by Equation (44). The information contained in this quadratic *may* lead to an analytical relationship which characterizes the bias, so that it can be compensated.. Alternatively, a simple *ad hoc* technique would be to conduct a performance analysis on the problem at hand and empirically determine the magnitude and direction of the bias term. This would probably lead to several different bias terms based on the current “mode of operation”, since the bias is likely to change significantly as the operational mode of the system changes.

The PBDM requires calculation of the Chi-Squared probability, P_{χ_j} , for each filter in the MMAE bank. The discussion on page 124 identified that P_{χ_j} should be selected greater than P_{χ_3} for the case in which filter 3 is assumed to match truth and $j > 3$. This was demonstrated for the case in which the parameter value being estimated is the measurement noise covariance \mathbf{R} . The rec-

ommendation is to validate that mismodeling in any of the other system matrices (\mathbf{Q}_{dj} , Φ_j , \mathbf{H}_j , and \mathbf{B}_{dj}) will result in this same attribute.

The parameter value search routine provided a means of simplifying the process of finding parameter values for the filters to meet the desired probability goals as described in Section 3.2.3.1. The recommendations on page 132 are (1) extend the search routine for the case in which the parameter vector is *not* a scalar and (2) incorporate both the magnitude of the *transformed* residual mean vector, $M_{\mu'}$, and the direction of the mean vector into the search routine.

Two recommendations associated with the Sheldon algorithm were proposed. First is to take the parameter values chosen by the Sheldon algorithm and set the one closest to $\hat{\mathbf{a}}_{\text{MMAE}}$ equal to $\hat{\mathbf{a}}_{\text{MMAE}}$ (see page 218). Alternatively, one could constrain the optimization such that one of the parameter values is fixed at $\hat{\mathbf{a}}_{\text{MMAE}}$. The motivation for these alternatives is to utilize the information provided through $\hat{\mathbf{a}}_{\text{MMAE}}$, namely the current best estimate of the true parameter value, to define one of the elemental filters in the bank in order to enhance the algorithm's performance.

The second recommendation is presented on page 222 as constraining the Sheldon algorithm by fixing the minimum and maximum filter-assumed parameter values in the optimization process. This would not only help the integration of the Sheldon algorithm with the density algorithm which provides the two endpoint parameter values, but would also prevent the severe underestimation of the parameter that existed in case 1 in which the true GPS noise interference level of $R_{GPS} = 18000 ft^2$ (versus the nominal noise level of $R_{GPS} = 9 ft^2$) was induced for 100 sec of the 200 sec flight profile. Recall that the Sheldon algorithm will not choose a parameter value for a filter equal to the maximum admissible parameter value, resulting in an upper bound of the parameter estimate that may be significantly less than the true parameter value. Therefore, it makes *good engineering sense* to fix the minimum and maximum filter-assumed parameter values. Alternatively, artificially increasing a_{max} prior to implementing Sheldon's algorithm such that a_j as computed by that algo-

rithm satisfies $a_J = a_{\max}$, would also prevent this severe parameter underestimation. However, this requires anticipating which artificial value of a_{\max} (in fact, establishing it iteratively) would generate the desired result and does not present any clear benefits over the method above of simply fixing a_J at a_{\max} in a *constrained* optimization.

The test cases simulated in Chapter 4 provide strong motivation for implementing these recommended changes to the Sheldon discretization in the context of a moving-bank MMAE. This is especially true for moving-bank algorithms that lead to contracted bank sizes which are precluded from moving to *all possible locations* in the admissible parameter space. The results show the unnecessary and potentially severe cost in terms of degraded parameter estimation, if the changes are not incorporated. The obvious objection to applying either of these two recommendations for the Sheldon algorithm is that *good engineering sense* does not guarantee optimality. Furthermore, the Sheldon algorithm is based on the concept of optimally choosing the parameter values for the filters, so applying *ad hoc* engineering may counter this optimality. Simulations followed by analysis will be necessary to determine any benefits resulting from these two recommendations.

The last recommendation is to apply modified blending of the parameter and state estimates when the breadth of the MMAE bank is particularly large. Modified blending is first discussed on page 264 as excluding the filters with a “low” probability from the blending process. The motivation is to prevent filters at or close to the probability lower bound from skewing the estimates high or low, which is more likely to occur when the breadth of the bank is large. This concept was shown to provide superior results under nominal flight conditions (no interference/jamming), but will likely provide similar benefits in the presence of interference/jamming and for other applications in general. This needs to be validated through simulations.

APPENDIX A - Derivation of Density Moments

This appendix presents the derivations of the first two moments of the *generalized* Chi-Squared density function, $f_\chi(c)$, in the presence of system mismodeling (see Sections 3.1.3 and 3.2). The quadratic of interest is given by

$$\chi_j = \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j$$

under the conditions that the parameter \mathbf{a}_t in the truth model is different from the parameter \mathbf{a}_j used as the basis of the j^{th} elemental filter within the MMAE. Explicitly, the residual $\mathbf{r}_j(t_i)$ in that j^{th} filter is given by $[\mathbf{z}(t_i) - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)]$, and the measurement $\mathbf{z}(t_i)$ is produced by the real world sensor, represented by the truth model based on the parameter value \mathbf{a}_t . The mean of the random variable χ_j is given by

$$\begin{aligned} E\{\chi_j\} &= E\{\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j\} = E\left\{tr\left[\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j\right]\right\} = E\left\{tr\left[(\mathbf{r}_j^T) (\mathbf{A}_j^{-1} \mathbf{r}_j)\right]\right\} \\ &= E\left\{tr\left[(\mathbf{A}_j^{-1} \mathbf{r}_j) (\mathbf{r}_j^T)\right]\right\} = E\left\{tr\left[\mathbf{A}_j^{-1} \mathbf{r}_j \mathbf{r}_j^T\right]\right\} = tr\left[E\left\{\mathbf{A}_j^{-1} \mathbf{r}_j \mathbf{r}_j^T\right\}\right] \\ &= tr\left[\mathbf{A}_j^{-1} E\left\{\mathbf{r}_j \mathbf{r}_j^T\right\}\right] \end{aligned} \quad (167)$$

Recall that the residuals, \mathbf{r}_j , are normally distributed with mean, $\boldsymbol{\mu}_j$, and that Hanlon [22] showed that the *actual* covariance matrix for the residuals is the *true* covariance matrix, \mathbf{A}_t , given by

$$\begin{aligned} \mathbf{A}_t &= E\left\{(\mathbf{r}_j - \boldsymbol{\mu}_j)(\mathbf{r}_j - \boldsymbol{\mu}_j)^T\right\} = E\left\{\mathbf{r}_j \mathbf{r}_j^T\right\} - \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T \\ \Rightarrow E\left\{\mathbf{r}_j \mathbf{r}_j^T\right\} &= \mathbf{A}_t + \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T \end{aligned} \quad (168)$$

Substituting Equation (168) into (167) gives the final expression for the mean as:

$$\mu_{\chi_j} = E\{\chi_j\} = tr\left[\mathbf{A}_j^{-1} (\mathbf{A}_t + \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T)\right] \quad (169)$$

The variance is found as

$$\begin{aligned} \sigma_{\chi_j}^2 &= E\left\{(\chi_j - \mu_{\chi_j})^2\right\} = E\left\{\left(\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j - \mu_{\chi_j}\right)^2\right\} \\ &= E\left\{tr\left(\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j - \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \mu_{\chi_j} - \mu_{\chi_j} \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j + \mu_{\chi_j}^2\right)\right\} \\ &= E\left\{tr\left(\mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j - 2\mathbf{A}_j^{-1} \mathbf{r}_j \mathbf{r}_j^T \mu_{\chi_j} + \mu_{\chi_j}^2\right)\right\} \end{aligned}$$

$$\begin{aligned}
&= E \left\{ \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \right\} - 2\mu_{\chi_j} \text{tr} \left(\mathbf{A}_j^{-1} E \left\{ \mathbf{r}_j \mathbf{r}_j^T \right\} \right) + \mu_{\chi_j}^2 \\
&= E \left\{ \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \right\} - 2\mu_{\chi_j}^2 + \mu_{\chi_j}^2
\end{aligned} \tag{170}$$

where the last equality uses Equation (167). From Appendix E of Maybeck [43], the following relationship is obtained:

$$E \left\{ \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \mathbf{r}_j^T \mathbf{A}_j^{-1} \mathbf{r}_j \right\} = \text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \right) \text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \right) + 2\text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \mathbf{A}_j^{-1} \mathbf{A}_t \right) \tag{171}$$

Now by substituting Equations (171) and (169) into (170), the final expression for the variance is found as:

$$\begin{aligned}
\sigma_{\chi_j}^2 &= \text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \right) \text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \right) + 2\text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \mathbf{A}_j^{-1} \mathbf{A}_t \right) - 2\mu_{\chi_j}^2 + \mu_{\chi_j}^2 \\
&= \text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \right) \text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \right) + 2\text{tr} \left(\mathbf{A}_j^{-1} \mathbf{A}_t \mathbf{A}_j^{-1} \mathbf{A}_t \right) - \mu_{\chi_j}^2
\end{aligned} \tag{172}$$

Notice that the expressions in Equations (169) and (172) are functions of μ_j , \mathbf{A}_j and \mathbf{A}_t , which are readily available from the MMAE structure and assumed system models.

APPENDIX B - Numerical Integration

This appendix introduces the development of a new numerical integration method denoted the "Modified Simpson's Rules". The derivation is heavily based on the method used to derive the well-known 3- and 4-point Simpson's rules used for numerical integration in one dimension [63]. The modified Simpson's rules are a more general form of the basic Simpson's rules, allowing the spacing between the discrete samples used in the integration to be non-uniform. This new development was motivated by the need to integrate a finite set of samples of a function in which the samples are not evenly spaced along the abscissa. Other methods considered include the trapezoid rule, curve fitting the data followed by analytic integrals, and Runge-Kutta methods. The modified Simpson's rules provided the best trade-off between numerical accuracy and computer processing time. Additionally, Runge-Kutta methods require a functional form of the integrand which is not available for problems with a finite set of samples of the function. The 3-point modified Simpson's rule is derived first, followed by the 4-point modified Simpson's rule.

Consider Figure 116 and begin by approximating the integral of the function $f(x)$ by

$$\begin{aligned}\int_{x_0}^{x_2} f(x)dx &= h_1 w_0 f_0 + \frac{h_1}{h_1 + h_2} w_1 f_1 + \frac{h_2}{h_1 + h_2} w_1 f_1 + h_2 w_2 f_2 \\ &= h_1 w_0 f_0 + \frac{h_1 + h_2}{h_1 + h_2} w_1 f_1 + h_2 w_2 f_2 \\ &= h_1 w_0 f_0 + w_1 f_1 + h_2 w_2 f_2\end{aligned}\tag{173}$$

where

$$\begin{aligned}x_i &= \text{discrete locations of functional values, } i = 0, 1, 2 \\ f_i &= \text{function evaluated at discrete locations } x_i, i = 0, 1, 2 \\ w_i &= \text{weighting values to be determined, } i = 0, 1, 2 \\ h_j &= \text{interval spacing between discrete locations, } j = 1, 2\end{aligned}$$

Note that the form of the first equality in Equation (173) is motivated by the standard 3-point Simpson's rule [63].

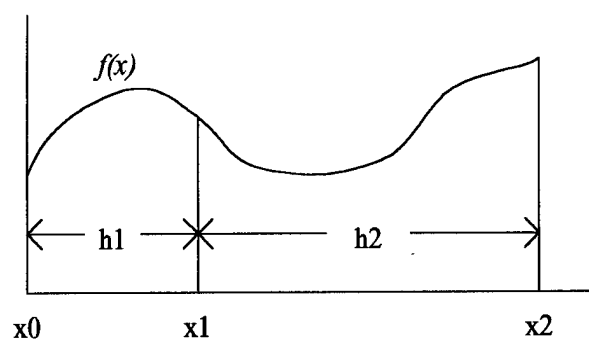


Figure 116. Non-Uniform Spacing for Modified Simpson's Rules

Now let

$$\begin{aligned} x_0 &= 0 \\ x_1 &= x_0 + h_1 = h_1 \\ x_2 &= x_1 + h_2 = h_1 + h_2 \end{aligned}$$

Next let $f(x)$ take on 3 expressions for which an exact integral is known, such as $f(x) = 1, x, x^2$.

This will lead to 3 equations for the 3 unknown weights w_i . Start with $f(x) = 1$ which results in

$$\begin{aligned} \int_{x_0}^{x_2} 1 dx &= h_1 w_0 + w_1 + h_2 w_2 \\ x_2 - x_0 &= h_1 w_0 + w_1 + h_2 w_2 \\ h_1 + h_2 - 0 &= h_1 w_0 + w_1 + h_2 w_2 \\ 1 &= \left(\frac{h_1}{h_1 + h_2} \right) w_0 + \left(\frac{1}{h_1 + h_2} \right) w_1 + \left(\frac{h_2}{h_1 + h_2} \right) w_2 \end{aligned} \quad (174)$$

Next let $f(x) = x$ which results in

$$\begin{aligned}
\int_{x_0}^{x_2} x dx &= h_1 w_0 x_0 + w_1 x_1 + h_2 w_2 x_2 \\
\frac{x_2^2 - x_0^2}{2} &= h_1 w_1 + h_2 w_2 (h_1 + h_2) \\
\frac{(h_1 + h_2)^2}{2} &= h_1 w_1 + h_2 (h_1 + h_2) w_2 \\
\frac{1}{2} &= \left(\frac{h_1}{(h_1 + h_2)^2} \right) w_1 + \left(\frac{h_2}{h_1 + h_2} \right) w_2
\end{aligned} \tag{175}$$

Finally, let $f(x) = x^2$ which results in

$$\begin{aligned}
\int_{x_0}^{x_2} x^2 dx &= h_1 w_0 x_0^2 + w_1 x_1^2 + h_2 w_2 x_2^2 \\
\frac{x_2^3 - x_0^3}{3} &= h_1^2 w_1 + h_2 w_2 (h_1 + h_2)^2 \\
\frac{(h_1 + h_2)^3}{3} &= h_1^2 w_1 + h_2 (h_1 + h_2)^2 w_2 \\
\frac{1}{3} &= \left(\frac{h_1^2}{(h_1 + h_2)^3} \right) w_1 + \left(\frac{h_2}{h_1 + h_2} \right) w_2
\end{aligned} \tag{176}$$

Combining Equations (174), (175) and (176) as a 3-by-3 system of equations permits the weights

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$$

to be found through a simple matrix inversion. Specifically,

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \frac{h_1}{h_1 + h_2} & \frac{1}{h_1 + h_2} & \frac{h_2}{h_1 + h_2} \\ 0 & \frac{h_1}{(h_1 + h_2)^2} & \frac{h_2}{h_1 + h_2} \\ 0 & \frac{h_1^2}{(h_1 + h_2)^3} & \frac{h_2}{h_1 + h_2} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \end{bmatrix} \tag{177}$$

which can be efficiently implemented in software by recognizing the often repeated term $h_1 + h_2$.

The 4-point modified Simpson's rule is derived in the same manner as above with the approximation of the integral of $f(x)$ given by

$$\int_{x_0}^{x_3} f(x) dx = h_1 w_0 f_0 + w_1 f_1 + w_2 f_2 + h_3 w_3 f_3 \quad (178)$$

Now let $f(x) = 1, x, x^2, x^3$, and this will lead to 4 equations for the 4 unknown weights w_i . The final result is

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} \frac{h_1}{h_1+h_2+h_3} & \frac{1}{h_1+h_2+h_3} & \frac{1}{h_1+h_2+h_3} & \frac{h_3}{h_1+h_2+h_3} \\ 0 & \frac{h_1}{(h_1+h_2+h_3)^2} & \frac{h_1+h_2}{(h_1+h_2+h_3)^2} & \frac{h_3}{h_1+h_2+h_3} \\ 0 & \frac{h_1^2}{(h_1+h_2+h_3)^3} & \frac{(h_1+h_2)^2}{(h_1+h_2+h_3)^3} & \frac{h_3}{h_1+h_2+h_3} \\ 0 & \frac{h_1^3}{(h_1+h_2+h_3)^4} & \frac{(h_1+h_2)^3}{(h_1+h_2+h_3)^4} & \frac{h_3}{h_1+h_2+h_3} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \\ \frac{1}{4} \end{bmatrix} \quad (179)$$

The 3- and 4-point rules are both required to account for data sets with even and odd numbers of samples. Notice that at least three samples must be available to implement the 3-point rule. The length of the data set is first checked, and the 3-point rule is applied repeatedly and exclusively if the data set has an odd number of samples. However, if there is an even number of samples being integrated, then the 3-point rule is applied repeatedly for all but the last three samples followed by a single implementation of the 4-point rule on the last four samples. Alternatively, the 4-point rule could be given precedence and applied repeatedly, followed by zero, one or two implementations of the 3-point rule on the last several samples as needed. Table 9 shows three examples to illustrate the combinations of 3- and 4-point Simpson's or Modified Simpson's rules. For example, with 10 samples, either apply three 3-point rules followed by one 4-point rule *or* apply three 4-point rules

and no 3-point rules. The advantage of giving 3-point rules precedence over 4-point rules is fewer inversions of the 4-by-4 matrix in Equation (179) versus the 3-by-3 matrix in Equation (177). There is a greater chance of ill-conditioning with the 4-by-4 inversions versus the 3-by-3 inversions.

Table 9. Example Combinations of Simpson's Rule

# Samples	3-point / 4-point	4-point / 3-point
10	3 / 1	3 / 0
11	5 / 0	2 / 2
12	4 / 1	3 / 1

APPENDIX C - Equations for the Mean of the Residual

Hanlon [22] developed equations for the mean of the residual of a Kalman filter given three mismodeling scenarios. Hanlon defined mismodeling as variations in the system models from the true system models through the following equations:

$$\Delta \mathbf{B}_j = \mathbf{B}_t - \mathbf{B}_j \rightarrow \mathbf{B}_j = \mathbf{B}_t - \Delta \mathbf{B}_j$$

$$\Delta \mathbf{H}_j = \mathbf{H}_t - \mathbf{H}_j \rightarrow \mathbf{H}_j = \mathbf{H}_t - \Delta \mathbf{H}_j$$

$$\Delta \Phi_j = \Phi_t - \Phi_j \rightarrow \Phi_j = \Phi_t - \Delta \Phi_j$$

The error between each elemental Kalman filter state estimate and true state is defined as

$$\epsilon_j(t_i^+) = \mathbf{x}_t(t_i) - \hat{\mathbf{x}}_j(t_i^+) \quad (180)$$

Assume that until a certain point in time, the Kalman filter model matched the true system model.

This time is identified as t_f and up until this time, all of the modeling errors are zero, $\epsilon_j(t_i^+) = \mathbf{0}$,

and

$$\left. \begin{array}{l} \mathbf{x}_t(t_i^+) = \hat{\mathbf{x}}_j(t_i^+) \\ \mathbf{x}_t(t_i^-) = \hat{\mathbf{x}}_j(t_i^-) \end{array} \right\} \quad \forall t_i < t_f$$

For each mismodeling case, a pair of equations is needed to calculate the mean of the residual. First, an iterative equation for the mean of the state estimate error is given by

$$E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_i^+) \} = \begin{cases} \mathbf{0} & , \quad \text{for } t_i \leq t_f \\ \begin{aligned} & [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \Phi_j - \mathbf{K}_j \Delta \mathbf{H}_j \Phi_j] \hat{\mathbf{x}}_j(t_f^+) \\ & + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \mathbf{B}_j - \mathbf{K}_j \Delta \mathbf{H}_j \mathbf{B}_j] \mathbf{u}(t_f) \\ & (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} \\ & + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \Phi_j - \mathbf{K}_j \Delta \mathbf{H}_j \Phi_j] \hat{\mathbf{x}}_j(t_{i-1}^+) \\ & + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \mathbf{B}_j - \mathbf{K}_j \Delta \mathbf{H}_j \mathbf{B}_j] \mathbf{u}(t_{i-1}) \end{aligned} & , \quad \text{for } t_i = t_{f+1} \\ \begin{aligned} & + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \Phi_j - \mathbf{K}_j \Delta \mathbf{H}_j \Phi_j] \hat{\mathbf{x}}_j(t_{i-1}^+) \\ & + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \mathbf{B}_j - \mathbf{K}_j \Delta \mathbf{H}_j \mathbf{B}_j] \mathbf{u}(t_{i-1}) \end{aligned} & , \quad \text{for } t_i > t_{f+1} \end{cases} \quad (181)$$

where $E_{\mathbf{Z}(t_{i-1})} \{ \cdot \} \triangleq E \{ \cdot | \mathbf{Z}(t_{i-1}) = \mathbf{Z}_{i-1} \}$, i.e., the *conditional* expectation, conditioned on the previous measurement history. Second, a running calculation of the mean state estimate error is

used to compute the residual mean, $\mu_j = E_{\mathbf{Z}(t_{i-1})} \{\mathbf{r}_j(t_i)\}$, via the following equation:

$$\begin{aligned} E_{\mathbf{Z}(t_{i-1})} \{\mathbf{r}_j(t_i)\} &= \mathbf{H}_t \Phi_t E_{\mathbf{Z}(t_{i-1})} \{\epsilon_j(t_{i-1}^+)\} \\ &\quad + (\mathbf{H}_t \Delta \Phi_j + \Delta \mathbf{H}_j \Phi_t - \Delta \mathbf{H}_j \Delta \Phi_j) \hat{\mathbf{x}}_j(t_{i-1}^+) \\ &\quad + (\mathbf{H}_t \Delta \mathbf{B}_j + \Delta \mathbf{H}_j \mathbf{B}_t - \Delta \mathbf{H}_j \Delta \mathbf{B}_j) \mathbf{u}(t_{i-1}) \end{aligned} \quad (182)$$

Equations (181) and (182) can be simplified for each of the following cases in which only one type of mismodeling is assumed. These cases could be extended for combinations of mismodeling such as $\Delta \mathbf{B}_j \neq 0$ and $\Delta \mathbf{H}_j \neq 0$.

C.1 Mismodeled Input Matrix

Given that

$$\Delta \mathbf{B}_j \neq 0$$

Equation (181) reduces to

$$E_{\mathbf{Z}(t_{i-1})} \{\epsilon_j(t_i^+)\} = \begin{cases} 0 & , \quad \text{for } t_i \leq t_f \\ (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \mathbf{B}_j \mathbf{u}(t_f) & , \quad \text{for } t_i = t_{f+1} \\ (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Phi_t E_{\mathbf{Z}(t_{i-1})} \{\epsilon_j(t_{i-1}^+)\} \\ \quad + (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \mathbf{B}_j \mathbf{u}(t_{i-1}) & , \quad \text{for } t_i > t_{f+1} \end{cases}$$

and Equation (182) reduces to

$$E_{\mathbf{Z}(t_{i-1})} \{\mathbf{r}_j(t_i)\} = \mathbf{H}_t \Phi_t E_{\mathbf{Z}(t_{i-1})} \{\epsilon_j(t_{i-1}^+)\} + \mathbf{H}_t \Delta \mathbf{B}_j \mathbf{u}(t_{i-1})$$

C.2 Mismodeled Output Matrix

Given that

$$\Delta \mathbf{H}_j \neq 0$$

Equation (181) reduces to

$$E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_i^+) \} = \begin{cases} 0 & , \quad \text{for } t_i \leq t_f \\ -\mathbf{K}_j \Delta \mathbf{H}_j \Phi_j \hat{\mathbf{x}}_j(t_f^+) - \mathbf{K}_j \Delta \mathbf{H}_j \mathbf{B}_j \mathbf{u}(t_f) & , \quad \text{for } t_i = t_{f+1} \\ (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} & \\ -\mathbf{K}_j \Delta \mathbf{H}_j \Phi_j \hat{\mathbf{x}}_j(t_{i-1}^+) - \mathbf{K}_j \Delta \mathbf{H}_j \mathbf{B}_j \mathbf{u}(t_{i-1}) & , \quad \text{for } t_i > t_{f+1} \end{cases}$$

and Equation (182) reduces to

$$E_{\mathbf{Z}(t_{i-1})} \{ \mathbf{r}_j(t_i) \} = \mathbf{H}_t \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} + \Delta \mathbf{H}_j \Phi_t \hat{\mathbf{x}}_j(t_{i-1}^+) + \Delta \mathbf{H}_j \mathbf{B}_t \mathbf{u}(t_{i-1})$$

C.3 Mismodeled State Transition Matrix

Given that

$$\Delta \Phi_j \neq 0$$

Equation (181) reduces to

$$E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_i^+) \} = \begin{cases} 0 & , \quad \text{for } t_i \leq t_f \\ (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \Phi_j \hat{\mathbf{x}}_j(t_f^+) & , \quad \text{for } t_i = t_{f+1} \\ (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} & \\ + (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \Phi_j \hat{\mathbf{x}}_j(t_{i-1}^+) & , \quad \text{for } t_i > t_{f+1} \end{cases}$$

and Equation (182) reduces to

$$E_{\mathbf{Z}(t_{i-1})} \{ \mathbf{r}_j(t_i) \} = \mathbf{H}_t \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} + \mathbf{H}_t \Delta \Phi_j \hat{\mathbf{x}}_j(t_{i-1}^+)$$

C.4 Mismodeling Measurement or Dynamics Noise Covariances

Hanlon did not explicitly derive expressions for the mean of the state estimate error or the mean of the residual in the presence of mismodeled measurement or dynamics noise covariances.

However, his work is easily extended for these cases by letting:

$$\Delta \mathbf{R}_j = \mathbf{R}_t - \mathbf{R}_j \rightarrow \mathbf{R}_j = \mathbf{R}_t - \Delta \mathbf{R}_j$$

$$\Delta \mathbf{Q}_{dj} = \mathbf{Q}_{dt} - \mathbf{Q}_{dj} \rightarrow \mathbf{Q}_{dj} = \mathbf{Q}_{dt} - \Delta \mathbf{Q}_{dj}$$

Begin by expressing the residual vector in terms of the mismodeled system matrices:

$$\begin{aligned} \mathbf{r}_j(t_i) &\triangleq \mathbf{z}(t_i) - \mathbf{H}_j \hat{\mathbf{x}}_j(t_i^-) = [\mathbf{H}_t \mathbf{x}_t(t_i) + \mathbf{v}_t(t_i)] - \mathbf{H}_j \hat{\mathbf{x}}_j(t_i^-) \\ &= \mathbf{H}_t [\Phi_t \mathbf{x}_t(t_{i-1}) + \mathbf{B}_t \mathbf{u}(t_{i-1}) + \mathbf{G}_t \mathbf{w}_{dt}(t_{i-1})] + \mathbf{v}_t(t_i) - \mathbf{H}_j [\Phi_j \hat{\mathbf{x}}_j(t_{i-1}^+) + \mathbf{B}_j \mathbf{u}(t_{i-1})] \\ &= \mathbf{H}_t \Phi_t \mathbf{x}_t(t_{i-1}) - \mathbf{H}_j \Phi_j \hat{\mathbf{x}}_j(t_{i-1}^+) + [\mathbf{H}_t \mathbf{B}_t - \mathbf{H}_j \mathbf{B}_j] \mathbf{u}(t_{i-1}) + \mathbf{H}_t \mathbf{G}_t \mathbf{w}_{dt}(t_{i-1}) + \mathbf{v}_t(t_i) \\ &= \mathbf{H}_t \Phi_t \mathbf{x}_t(t_{i-1}) - (\mathbf{H}_t - \Delta \mathbf{H}_j) (\Phi_t - \Delta \Phi_j) \hat{\mathbf{x}}_j(t_{i-1}^+) \\ &\quad + [\mathbf{H}_t \mathbf{B}_t - (\mathbf{H}_t - \Delta \mathbf{H}_j) (\mathbf{B}_t - \Delta \mathbf{B}_j)] \mathbf{u}(t_{i-1}) + \mathbf{H}_t \mathbf{G}_t \mathbf{w}_{dt}(t_{i-1}) + \mathbf{v}_t(t_i) \\ &= \mathbf{H}_t \Phi_t \epsilon_j(t_{i-1}^+) + (\mathbf{H}_t \Delta \Phi_j + \Delta \mathbf{H}_j \Phi_t - \Delta \mathbf{H}_j \Delta \Phi_j) \hat{\mathbf{x}}_j(t_{i-1}^+) \\ &\quad + (\mathbf{H}_t \Delta \mathbf{B}_j + \Delta \mathbf{H}_j \mathbf{B}_t - \Delta \mathbf{H}_j \Delta \mathbf{B}_j) \mathbf{u}(t_{i-1}) + \mathbf{H}_t \mathbf{G}_t \mathbf{w}_{dt}(t_{i-1}) + \mathbf{v}_t(t_i) \end{aligned} \quad (183)$$

where $\mathbf{w}_{dt}(t_i)$ is the discrete-time zero-mean white Gaussian dynamics driving noise vector for the truth model and $\mathbf{v}_t(t_i)$ is the discrete-time zero-mean white Gaussian measurement noise vector for the truth model. Notice the absence of $\Delta \mathbf{R}_j$ and $\Delta \mathbf{Q}_{dj}$ explicitly in Equation (183), realizing that they will affect the explicitly shown terms, $\epsilon_j(t_i^+)$ and $\hat{\mathbf{x}}_j(t_i^+)$. Next take the expectation of this residual expression to find the mean of the residual:

$$\begin{aligned} E_{\mathbf{Z}(t_{i-1})} \{\mathbf{r}_j(t_i)\} &= E_{\mathbf{Z}(t_{i-1})} \{ \mathbf{H}_t \Phi_t \epsilon_j(t_{i-1}^+) + (\mathbf{H}_t \Delta \Phi_j + \Delta \mathbf{H}_j \Phi_t - \Delta \mathbf{H}_j \Delta \Phi_j) \hat{\mathbf{x}}_j(t_{i-1}^+) \\ &\quad + (\mathbf{H}_t \Delta \mathbf{B}_j + \Delta \mathbf{H}_j \mathbf{B}_t - \Delta \mathbf{H}_j \Delta \mathbf{B}_j) \mathbf{u}(t_{i-1}) + \mathbf{H}_t \mathbf{G}_t \mathbf{w}_{dt}(t_{i-1}) + \mathbf{v}_t(t_i) \} \\ &= \mathbf{H}_t \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} \\ &\quad + (\mathbf{H}_t \Delta \Phi_j + \Delta \mathbf{H}_j \Phi_t - \Delta \mathbf{H}_j \Delta \Phi_j) \hat{\mathbf{x}}_j(t_{i-1}^+) \\ &\quad + (\mathbf{H}_t \Delta \mathbf{B}_j + \Delta \mathbf{H}_j \mathbf{B}_t - \Delta \mathbf{H}_j \Delta \mathbf{B}_j) \mathbf{u}(t_{i-1}) \end{aligned}$$

where the last simplification results from $E_{\mathbf{Z}(t_{i-1})} \{\mathbf{w}_{dt}(t_i)\} = \mathbf{0}$ and $E_{\mathbf{Z}(t_{i-1})} \{\mathbf{v}_t(t_i)\} = \mathbf{0}$ (i.e., zero means). This result is exactly Equation (182), indicating that this expression still holds in the presence of mismodeled measurement or dynamics noise covariances. Similarly, Equation (181) is duplicated by taking the conditional expectation of Equation (180) and stepping back one data

sample in time:

$$\begin{aligned}
E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_i^+) \} &= (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} \\
&\quad + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \Phi_j - \mathbf{K}_j \Delta \mathbf{H}_j \Phi_j] \hat{\mathbf{x}}_j(t_{i-1}^+) \\
&\quad + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \mathbf{B}_j - \mathbf{K}_j \Delta \mathbf{H}_j \mathbf{B}_j] \mathbf{u}(t_{i-1}) \\
&\quad + (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \mathbf{G}_t E_{\mathbf{Z}(t_{i-1})} \{ \mathbf{w}_{dt}(t_{i-1}) \} \\
&\quad - \mathbf{K}_j E_{\mathbf{Z}(t_{i-1})} \{ \mathbf{v}_t(t_i) \} \\
&= (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} \\
&\quad + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \Phi_j - \mathbf{K}_j \Delta \mathbf{H}_j \Phi_j] \hat{\mathbf{x}}_j(t_{i-1}^+) \\
&\quad + [(\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Delta \mathbf{B}_j - \mathbf{K}_j \Delta \mathbf{H}_j \mathbf{B}_j] \mathbf{u}(t_{i-1})
\end{aligned}$$

Furthermore, $\Delta \mathbf{B}_j = \mathbf{0}$, $\Delta \mathbf{H}_j = \mathbf{0}$ and $\Delta \Phi_j = \mathbf{0}$, resulting in the simplified expressions shown here.

$$E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_i^+) \} = \begin{cases} \mathbf{0} & , \text{ for } t_i \leq t_f \\ \mathbf{0} & , \text{ for } t_i = t_{f+1} \\ (\mathbf{I} - \mathbf{K}_j \mathbf{H}_t) \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} & , \text{ for } t_i > t_{f+1} \end{cases}$$

Which simplifies even further to

$$E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_i^+) \} = \mathbf{0}, \forall t_i$$

Similarly,

$$E_{\mathbf{Z}(t_{i-1})} \{ \mathbf{r}_j(t_i) \} = \mathbf{H}_t \Phi_t E_{\mathbf{Z}(t_{i-1})} \{ \epsilon_j(t_{i-1}^+) \} = \mathbf{0}, \forall t_i$$

This shows that the mean of the residual will be zero if the measurement or dynamics noise covariances alone are being mismodeled.

APPENDIX D - Model State Definitions and System Matrices

This appendix contains a tabular listing of the 13-state GPS/INS reduced-order model. The LN-93 error-state dynamics matrix F and the process noise matrix Q as provided by Litton are 93-by-93 arrays containing a large number of elements that are identically zero [25]. The *non-zero* elements of the Litton model that apply to the reduced-order 13-state model are included in Tables 10 through 15.

Table 10. Reduced-Order System Model States

State Number	State Symbol	Definition	LN-93 State	PLS State
1	$\delta\theta_x$	X-component of vector angle from true to computer frame	1	1
2	$\delta\theta_y$	Y-component of vector angle from true to computer frame	2	2
3	$\delta\theta_z$	Z-component of vector angle from true to computer frame	3	3
4	ϕ_x	X-component of vector angle from true to platform frame	4	4
5	ϕ_y	Y-component of vector angle from true to platform frame	5	5
6	ϕ_z	Z-component of vector angle from true to platform frame	6	6
7	δV_x	X-component of error in computed velocity	7	7
8	δV_y	Y-component of error in computed velocity	8	8
9	δV_z	Z-component of error in computed velocity	9	9
10	δh	Error in vehicle altitude above reference ellipsoid	10	10
11	δh_B	Total baro-altimeter correlated error	23	11
12	δPR_{Uclk}	GPS User clock bias	-	12
13	δD_{Uclk}	GPS User clock drift	-	13

Table 11. Elements of the Dynamics Submatrix $F_{(red)11}$

Element	Term	Element	Term
(1,3)	$-\rho_y$	(1,8)	$-C_{RY}$
(2,3)	ρ_x	(2,7)	C_{RX}
(3,1)	ρ_y	(3,2)	$-\rho_x$
(4,2)	$-\Omega_z$	(4,3)	Ω_y
(4,5)	ω_{in_z}	(4,6)	$-\omega_{in_y}$
(4,8)	$-C_{RY}$	(5,1)	Ω_z
(5,3)	$-\Omega_x$	(5,4)	$-\omega_{in_z}$
(5,6)	ω_{in_x}	(5,7)	C_{RX}
(6,1)	$-\Omega_y$	(6,2)	Ω_x
(6,4)	ω_{in_y}	(6,5)	$-\omega_{in_x}$
(7,1)	$-2V_y\Omega_y - 2V_z\Omega_z$	(7,2)	$2V_y\Omega_x$
(7,3)	$2V_z\Omega_y$	(7,5)	$-A_z$
(7,6)	A_y	(7,7)	$-V_zC_{RX}$
(7,8)	$2\Omega_z$	(7,9)	$-\rho_y - 2\Omega_y$
(8,1)	$2V_x\Omega_y$	(8,2)	$-2V_x\Omega_x - 2V_z\Omega_z$
(8,3)	$2V_z\Omega_y$	(8,4)	A_z
(8,6)	$-A_x$	(8,7)	$-2\Omega_z$
(8,8)	$-V_zC_{RY}$	(8,9)	$\rho_x + 2\Omega_x$
(9,1)	$2V_x\Omega_z$	(9,2)	$2V_y\Omega_z$
(9,3)	$-2V_y\Omega_y - 2V_x\Omega_x$	(9,4)	$-A_y$
(9,5)	A_x	(9,7)	$\rho_y + 2\Omega_y + V_xC_{RX}$
(9,8)	$-\rho_x - 2\Omega_x + V_yC_{RY}$	(9,10)	$2g_o/a$
(10,9)	1		

- $\rho_{x,y}$ = Components of angular rate, nav reference frame to earth-fixed frame
 $\Omega_{x,y,z}$ = Components of angular rate, earth-fixed frame to inertial frame
 $\omega_{in_{x,y,z}}$ = Components of angular rate, nav reference frame to inertial frame
 $V_{x,y,z}$ = Components of vehicle velocity vector in earth-fixed coordinates
 $A_{x,y,z}$ = Components of specific force in the sensor reference frame
 $C_{RX,RY}$ = Components of earth spheroid inverse radii of curvature
 g_o = Equatorial gravity magnitude (32.08744 ft/sec²)
 a = Equatorial radius of the earth (6378388 m)

Table 12. Elements of the Dynamics Submatrix $F_{(red)12}$

Element	Term	Element	Term
(9,11)	k_2	(10,11)	k_1

Table 13. Elements of the Dynamics Submatrix $F_{(red)22}$

Element	Term
(11,11)	$-\beta_{\delta h_c}$

Table 14. Elements of Process Noise Submatrix $Q_{(red)11}$

Element	Term	Element	Term
(4,4)	$Q_{\eta_{b_x}}$	(7,7)	$Q_{\eta_{A_x}}$
(5,5)	$Q_{\eta_{b_y}}$	(8,8)	$Q_{\eta_{A_y}}$
(6,6)	$Q_{\eta_{b_z}}$	(9,9)	$Q_{\eta_{A_z}}$

Table 15. Elements of Process Noise Submatrix $Q_{(red)22}$

Element	Term
(11,11)	$2\beta_{\delta h_c} \sigma_{\delta h_c}^2$

- $k_{1,2}$ = Vertical channel gains, see LN-93 documentation [25] for equations
 $\beta_{\delta h_c}$ = Barometer inverse correlation time ($\tau = 10$ min; $\beta = \frac{1}{\tau}$)
 $Q_{\eta_{b_x,y,z}}$ = PSD value of gyro drift rate white noise ($6.25e-10 \frac{deg^2}{sec^3}$)
 $Q_{\eta_{A_x,y,z}}$ = PSD value of accelerometer white noise ($1.037e-7 \frac{ft^2}{sec^3}$)
 $\sigma_{\delta h_c}^2$ = Variance of barometric altimeter correlated noise ($10000 ft^2$)

APPENDIX E - Tabulated Performance Measures

The performance measures \hat{e}_{RMS}^a and \hat{e}_{RMS}^x are discussed in Chapter 4 and shown here in Tables 16 – 22. The definition of these performance measures is given in Section 4.4. Notice that the minimum measure values (i.e., the ones associated with the best performance) for each case are highlighted in boldface, and Chapter 5 will draw final comparisons of the results tabulated here.

Table 16. MMAE Blended Parameter Estimation Measure \hat{e}_{RMS}^a

Case	Fix	Den	Den/Exp	Sheldon	Sheldon/Exp	Prob	Den/Prob
1	203	80	85	193	295	78	58
2	71	146	115	150	100	192	182
3	128	110	118	82	52	114	149
4	227	189	198	262	282	217	215
5	239	284	179	582	293	253	276
6	34	34	34	34	34	34	34

- Fix - Fixed-Bank Algorithm
- Den - Density Algorithm
- Den/Exp - Density Algorithm with Expansion and Increased Delay
- Sheldon - Density Algorithm with Sheldon Discretization
- Sheldon/Exp - Density Algorithm with Sheldon Discretization, Expansion and Increased Delay
- Prob - Probability Algorithm
- Den/Prob - Density Algorithm with Probability Discretization

Table 17. MMAE Blended State Estimation Measure \hat{e}_{RMS}^x – State 1

Case	Fix	Den	Den/Exp	Sheldon	Sheldon/Exp	Prob	Den/Prob
1	3.463	3.045	3.248	3.047	3.284	4.008	2.858
2	3.206	3.064	3.084	3.361	3.043	3.206	2.874
3	3.274	2.526	2.794	2.835	2.685	3.13	2.544
4	4.508	4.319	4.433	4.504	4.541	4.684	4.544
5	3.546	3.312	3.415	2.571	3.291	3.777	2.803
6	1.171	1.171	1.171	1.171	1.171	1.171	1.171

Table 18. MMAE Blended State Estimation Measure \hat{e}_{RMS}^x – State 2

Case	Fix	Den	Den/Exp	Prob	Sheldon	Sheldon/Exp	Den/Prob
1	3.932	3.23	3.483	4.541	3.433	3.632	3.029
2	3.689	3.159	3.554	3.833	3.056	3.335	2.914
3	3.911	3.523	3.369	3.734	3.349	3.389	3.376
4	6.11	5.917	5.997	6.207	6.356	6.359	6.045
5	3.988	3.048	3.824	4.154	2.732	3.42	3.512
6	1.329	1.329	1.329	1.329	1.329	1.329	1.329

Table 19. MMAE Blended State Estimation Measure \hat{e}_{RMS}^x – State 3

Case	Fix	Den	Den/Exp	Sheldon	Sheldon/Exp	Prob	Den/Prob
1	4.145	4.044	4.132	4.046	4.197	4.261	4.336
2	4.105	5.137	4.32	4.506	4.293	4.264	4.822
3	4.102	5.143	4.251	4.761	4.269	4.192	4.639
4	4.224	4.811	4.524	4.819	4.385	4.235	5.122
5	5.345	6.018	5.427	5.268	5.701	5.43	5.872
6	2.648	2.648	2.648	2.648	2.648	2.648	2.648

Table 20. M³AE Final State Estimation Measure \hat{e}_{RMS}^x – State 1

Case	Fix	Den	Den/Exp	Sheldon	Sheldon/Exp	Prob	Den/Prob
1	2.637	2.426	2.427	2.519	2.693	2.515	2.386
2	2.434	2.407	2.417	2.372	2.522	2.637	2.507
3	2.477	2.406	2.45	2.371	2.399	2.48	2.388
4	4.306	4.477	4.315	4.501	4.753	4.34	4.632
5	3.194	3.334	3.123	2.717	3.308	3.333	3.344
6	0.785	0.785	0.785	0.785	0.785	0.785	0.785

Table 21. M³AE Final State Estimation Measure \hat{e}_{RMS}^x – State 2

Case	Fix	Den	Den/Exp	Sheldon	Sheldon/Exp	Prob	Den/Prob
1	3.353	3.007	3.08	3.199	3.434	3.183	2.935
2	3.1	3.047	3.043	3.009	3.097	3.127	2.912
3	3.25	3.042	3.094	3.108	3.083	3.264	3.112
4	6.322	6.398	6.214	6.67	6.722	6.381	6.474
5	3.811	4.047	3.794	3.016	3.796	3.55	3.83
6	0.858	0.858	0.858	0.858	0.858	0.858	0.858

Table 22. M³AE Final State Estimation Measure \hat{e}_{RMS}^x – State 3

Case	Fix	Den	Den/Exp	Sheldon	Sheldon/Exp	Prob	Den/Prob
1	4.061	4.049	4.059	4.049	4.048	4.092	4.044
2	4.045	4.036	4.048	4.037	4.038	4.068	4.041
3	4.044	4.034	4.045	4.033	4.032	4.047	4.039
4	4.804	4.806	4.805	4.8	4.801	4.804	4.796
5	5.405	5.412	5.409	5.41	5.425	5.385	5.397
6	4.322	4.322	4.322	4.322	4.322	4.322	4.322

Bibliography

- [1] Athans, M., et al. "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method - Part I: Equilibrium Flight," *IEEE Transactions on Automatic Control*, Vol. AC-22, No. 5, 768-780, October 1977.
- [2] Athans, M. and C. B. Chang. *Adaptive Estimation and Parameter Identification Using a Multiple Model Estimation Algorithm*. Technical Note 1976-28, Lexington, MA: Lincoln Lab., MIT, June 1976. ESD-TR-76-184.
- [3] Bar-Shalom, Yaakov and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques, and Software*. Boston, MA: Artech House, 1993.
- [4] Baram, Yoram. *Information, Consistent Estimation and Dynamic System Identification*. PhD dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, Cambridge, Massachusetts, November 1976.
- [5] Blom, Henk A. P. and Yaakov Bar-Shalom. "The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients," *IEEE Transactions on Automatic Control*, Vol. 33, No. 8, 780-783, August 1988.
- [6] Britting, Kenneth R. *Inertial Navigation Systems Analysis*. New York: Wiley-Interscience, 1971.
- [7] Britton, Ryan, L. *A Differential GPS-aided INS for Aircraft Landings*. MS thesis, AFIT/GE/ENG/95D-03, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1995.
- [8] Carlson, Neal A. "Federated Filter for Fault-Tolerant Integrated Navigation," *Aerospace Navigation Systems*, Advisory Group for Aerospace Research and Development, AGARDograph Article 331, Neuilly-Sur-Seine France, June 1995.
- [9] Carlson, Neal A. *Distributed Kalman Filter Architectures Phase II*. Draft Final Report, WL/AAAI, Air Force Wright Laboratory, Wright-Patterson AFB, OH, April 1995.
- [10] Carlson, Neal A. and Jr. C. M. Neily. *Distributed Kalman Filter Architectures*. Final Technical Report, AFWAL-TR-87-1181, Air Force Avionics Laboratory, Wright-Patterson AFB, OH, June 1987.
- [11] Clark, Curtis S. *Multiple Model Adaptive Estimation and Control Redistribution for the Vista F-16 During Partial Actuator Impairments*. MS thesis, AFIT/GE/ENG/97D-23, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1997.
- [12] Deckert, James C., et al. "F-8 DFBW Sensor Failure Identification Using Analytic Redundancy," *IEEE Transactions on Automatic Control*, Vol. AC-22, No. 5, 795-803, October 1977.
- [13] Desai, M., et al. "Dual Sensor Failure Identification Using Analytic Redundancy," *AGARD Lecture Series No. 82, ISBN 92-835-0160-8* March 1976.
- [14] Eide, Peter K. *Implementation and Demonstration of a Multiple Model Adaptive Estimator Failure Detection System for the F-16*. MS thesis, AFIT/GE/ENG/94D-06, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December

1994.

- [15] Eide, Peter K. and Peter S. Maybeck. "An MMAE Failure Detection System for the F-16," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 32, No. 3, 1125–1148, July 1996.
- [16] Gray, Robert A. *An Integrated GPS/INS/BARO and Radar Altimeter System for Aircraft Precision Approach Landings*. MS thesis, AFIT/GE/ENG/94D-13, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1994.
- [17] Griffin, Gordon C. *Control of a Large Space Structure Using Multiple Model Adaptive Estimation and Control Techniques*. MS thesis, AFIT/GE/ENG/94D-14, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1994.
- [18] Griffin, Gordon C. and Peter S. Maybeck. "MMAE/MMAC Techniques Applied to Large Space Structure Bending with Multiple Uncertain Parameters," *Proceeding of the 34th IEEE Conference on Decision and Control*, New Orleans, LA, 1153–1158, December 1995.
- [19] Gustafson, John A. *Control of a Large Flexible Space Structure Using Multiple Model Adaptive Algorithms*. MS thesis, AFIT/GE/ENG/91D-22, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1991.
- [20] Gustafson, John A. and Peter S. Maybeck. "Flexible Spacestructure Control Via Moving-Bank Multiple Model Algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 30, No. 3, 750–757, July 1994.
- [21] Hanlon, Peter D. *Failure Identification Using Multiple Model Adaptive Estimation for the LAMBDA Flight Vehicle*. MS thesis, AFIT/GE/ENG/92D-19, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1992.
- [22] Hanlon, Peter D. *Practical Implementation of Multiple Model Adaptive Estimation Using Neyman-Pearson Based Hypothesis Testing and Spectral Estimation Tools*. PhD dissertation, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, Wright-Patterson AFB, OH, September 1996.
- [23] Hentz, K. P. *Feasibility Analysis of Moving Bank Multiple Model Adaptive Estimation and Control Algorithms*. MS thesis, AFIT/EE/ENG/84D-32, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1984.
- [24] Honeywell Military Avionics Division, Minneapolis, MN. *Honeywell AN/APN-194 Pulse Radar Altimeter System*, June 1989. Honeywell Technical Description.
- [25] Knudsen, L. *Performance Accuracy (Truth Model/Error Budget) Analysis for the LN-93 Inertial Navigation System Inertial Navigation Unit*. Technical Report, 5500 Canoga Avenue, Woodland Hills, California 91365: Litton Guidance and Control Systems, January 1985. DID No. Di-S-21433 B/T: CDRL No. 1002.
- [26] Kyger, David W. *Reducing Lag in Virtual Displays Using Multiple Model Adaptive Estimation*. MS thesis, AFIT/GE/ENG/95D-11, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1995.
- [27] Kyger, David W. and Peter S. Maybeck. "Reducing Lag in Virtual Displays Using Multiple Model Adaptive Estimation." Accepted for publication in *IEEE Transactions on AES*,

AES-34, No. 4, October 1998.

- [28] Lahey Computer Systems, Inc., P.O. Box 6091 Incline Village, NV 89450. *Lahey Fortran 90 User's Guide* (Version 3.0 Edition), 1995.
- [29] Lainiotis, D. G. "Partitioning: A Unifying Framework for Adaptive Systems, I: Estimation," *Proceedings of the IEEE*, 64, 1182-1197, August 1976.
- [30] Lewantowicz, Z. H. and D. W. Keen. "Graceful Degradation of GPS/INS Performance with Fewer Than Four Satellites," *The Institute of Navigation, National Technical Meeting*, Phoenix, AZ, 269-275, January 1991.
- [31] Li, Xiao-Rong, et. al. "Multiple-Model Estimation with Variable Structure: Model-Group Switching Algorithm," *Proceedings of the 36th IEEE Conference on Decision and Control*, San Diego, CA, 3114-3119, December 1997.
- [32] Li, Xiao-Rong and Yaakov Bar-Shalom. "Multiple-Model Estimation with Variable Structure," *IEEE Transactions on Automatic Control*, Vol. 41, No. 4, 479-493, April 1996.
- [33] Libby, E. W. *Application of Sequence Comparison Methods to Multisensor Data Fusion and Target Recognition*. PhD dissertation, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, June 1993.
- [34] Libby, E. W. and Peter S. Maybeck. "Sequence Comparison Techniques for Multisensor Data Fusion and Target Recognition," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 32, No. 1, 52-65, January 1996.
- [35] Lund, Elvind J. *On-line Discrimination and Estimation in Multiple Regime Systems*. PhD dissertation, Division of Engineering Cybernetics, The Norwegian Institute of Technology University of Trondheim, Trondheim, Norway, June 1992.
- [36] Lund, Elvind J., et al. "Multiple Model Estimation with Inter-Residual Distance Feedback," *Modeling, Identification and Control*, Vol. 13, No. 3, 127-140, 1992.
- [37] Magill, D. T. "Optimal Adaptive Estimation of Sampled Stochastic Processes," *IEEE Transactions on Automatic Control*, AC-10, No. 5, 434-439, 1965.
- [38] Martin, Daniel P. and Neal A. Carlson. *Distributed Model Adaptive Estimation Phase I Final Report*. Final Report, TR-91-003, WL/AAAN-2, Air Force Wright Laboratory, Wright-Patterson AFB, OH, August 1991.
- [39] MathSoft, Inc., 101 Main Street Cambridge, MA, 02142. *MATHCAD User's Guide* (Version 6.0 plus Edition), September 1995.
- [40] The MathWorks, Inc., Natick, MA. *Optimization Toolbox for Use with MATLAB (registered trademark)* (Version 4.2c Edition), December 1992.
- [41] The MathWorks, Inc., Natick, MA. *MATLAB (registered trademark)* (Version 4.2c Edition), November 1994.
- [42] Maybeck, Peter S. "Adaptive Tracking of Maneuvering Targets Based on IR Image Data," *AGARD Lecture Series No. 166*, published in London, England, 7, 1-18 July 1989.
- [43] Maybeck, Peter S. *Combined State and Parameter Estimation for On-Line Applications*. PhD dissertation, Massachusetts Institute of Technology, Charles Stark Draper Laboratory,

Cambridge, MA, 02139, February 1972.

- [44] Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, I. New York: Academic Press, Inc., 1979. Republished, Arlington, VA: Navtech, 1994.
- [45] Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, II. New York: Academic Press, Inc., 1982. Republished, Arlington, VA: Navtech, 1994.
- [46] Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, III. New York: Academic Press, Inc., 1982.
- [47] Maybeck, Peter S. "Moving-Bank Multiple Model Adaptive Estimation and Control Algorithms: An Evaluation," *Control and Dynamics Systems*, Vol. 31, 1989. Edited by C. T. Leondes, Academic Press, NY.
- [48] Maybeck, Peter S. and Peter D. Hanlon. "Performance Enhancement of a Multiple Model Adaptive Estimator," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 31, No. 4, 1240–1254, October 1995.
- [49] Maybeck, Peter S. and K. P. Hentz. "Investigation of Moving-Bank Multiple Model Adaptive Algorithms," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 1, 90–96, January–February 1987.
- [50] Maybeck, Peter S. and Richard D. Stevens. "Reconfigurable Flight Control Via Multiple Model Adaptive Control Methods," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 27, No. 3, 470–479, May 1991.
- [51] Menke, Timothy E. *Multiple Model Adaptive Estimation Applied to the VISTA F-16 with Actuator and Sensor Failures*. MS thesis, AFIT/GE/ENG/92J-01, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, June 1992.
- [52] Menke, Timothy E. and Peter S. Maybeck. "Sensor/Actuator Failure Detection in the VISTA F-16 by Multiple Model Adaptive Estimation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 31, No. 4, 1218–1228, October 1995.
- [53] Miller, Mikel M. *Modified Multiple Model Adaptive Estimation for Simultaneous Parameter and State Estimation*. PhD dissertation, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, March 1998.
- [54] Mosle, William B. *Detection, Isolation, and Recovery of Failures in an Integrated Navigation System*. MS thesis, AFIT/GE/ENG/93D-28, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.
- [55] Muravez, Randall J. *Multiple Model Adaptive Estimation and Prediction with the Harmonically Balanced Kalman Filter Bank*. MS thesis, California Polytechnic University, Pomona, CA, December 1989.
- [56] Musick, Stanton H. *PROFGEN – A Computer Program for Generating Flight Profiles*. Technical Report, Wright-Patterson AFB, OH: Air Force Avionics Laboratory, November 1976. AFAL-TR-76-247.
- [57] Musick, Stanton H. and Neil Carlson. *User's Manual for a Multimode Simulation for Optimal Filter Evaluation (MSOFE)*. Air Force Avionics Laboratory, Wright-Patterson AFB, OH, April 1990. AFWAL-TR-88-1136.

- [58] Negast, William J. *Incorporation of Differential Global Positioning System Measurements Using an Extended Kalman Filter for Improved Reference System Performance*. MS thesis, AFIT/GE/ENG/91D-41, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1991.
- [59] Nielsen, Robert L. *Development of a Performance Evaluation Tool (MMSOFE) for Detection of Failures with Multiple Model Adaptive Estimation (MMAE)*. MS thesis, AFIT/GE/ENG/93S-37, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.
- [60] Nielsen, Robert L. and Stanton H. Musick. *MMSOFE – Multiple Model Simulation for Optimal Filter Evaluation*. Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.
- [61] Oxley, Mark E., October 1997. Interview with Capt. Juan R. Vasquez, USAF.
- [62] Pinson, J. C. "Inertial Guidance for Cruise Vehicles," in *Guidance and Control of Aerospace Vehicles* (C. T. Leondes, ed.). McGraw-Hill, NY, 1963.
- [63] Press, William H., et al. *Numerical Recipes*. New York: Cambridge University Press, 1986.
- [64] Riggins, Robert N. Jr. *Detection and Isolation of Plant Failures in Dynamic Systems*. PhD dissertation, University of Michigan, 1991.
- [65] Scharf, Louis L. *Statistical Signal Processing*. Addison-Wesley Publishing Company, 1991.
- [66] Schiller, Gregory J. *Control of a Large Space Structure Using Multiple Model Adaptive Estimation and Control Techniques*. MS thesis, AFIT/GE/ENG/93D-02, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.
- [67] Schiller, Gregory J. and Peter S. Maybeck. "Space Structure Control Using Multiple Model Adaptive Estimation and Control," *Proceeding of the 13th IEAC Symposium Automatic Control in Aerospace-Aerospace Control '94*, September 1994.
- [68] Sheldon, Stuart N. *An Optimal Parameter Discretization Strategy for Multiple Model Adaptive Estimation and Control*. PhD dissertation, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1989.
- [69] Sheldon, Stuart N. and Peter S. Maybeck. "An Optimizing Design Strategy for Multiple Model Adaptive Estimation and Control," *IEEE Transactions on Automatic Control*, Vol. 38, No. 4, 651–654, April 1993.
- [70] Stepaniak, Michael J. *Multiple Model Adaptive Control of the VISTA F-16*. MS thesis, AFIT/GE/ENG/95D-26, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1995.
- [71] Stepaniak, Michael J. and Peter S. Maybeck. "MMAE-Based Control Redistribution Applied to the VISTA F-16." Accepted for publication in *IEEE Transactions on AES*, AES-34, No. 4, October 1998.
- [72] Stevens, Richard D. *Characterization of a Reconfigurable Multiple Model Adaptive Controller Using a STOL F-15 Model*. MS thesis, AFIT/GE/ENG/89D-52, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December

1989.

- [73] Strang, Gilbert. *Linear Algebra and Its Applications*. New York: Academic Press, Inc., 1976.
- [74] Van Trees, H. L. *Detection, Estimation and Modulation Theory*. New York: Wiley and Sons, 1968.
- [75] Vasquez, Juan R. *Detection of Spoofing, Jamming, or Failure of a Global Positioning System (GPS)*. MS thesis, AFIT/GE/ENG/92D-37, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1992.
- [76] Wald, Abraham. *Sequential Analysis*. New York: John Wiley and Sons, Inc., 1947.
- [77] White, Nathan A. *MMAE Detection of Interference/Jamming and Spoofing in a DGPS-Aided INS*. MS thesis, AFIT/GE/ENG/96D-21, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1996.
- [78] White, N. A., P. S. Maybeck and S.L. DeVilbiss. "Detection of Interference/Jamming and Spoofing in a DGPS-Aided Inertial System." Accepted for publication in *IEEE Transactions on AES*, AES-34, No. 4, October 1998.
- [79] Willsky, A. S. "A Survey of Design Methods for Failure Detection in Dynamic Systems," *Automatica*, No. 12, 601-611, December 1976.
- [80] Willsky, A. S. and H. Jones. *A Generalized Likelihood Ratio Approach to State Estimation in Linear Systems Subject to Abrupt Changes*. Technical Report, Reading, MA: The Analytic Sciences Corporation, 1976. Under USAF Contract No. F04701-74-C-0095.

Vita

Captain Juan Roberto Vasquez was born 1 Jun 1965 on Chanute AFB, Rantoul, Illinois. He has been in the Air Force since birth but did not enter active duty until 1988 after being commissioned through the Reserve Officer Training Program (ROTC) at Oklahoma State University (OSU). His life as a military dependent has sent him from England to Oklahoma where he graduated from Choctaw High School in 1983. An Air Force ROTC scholarship aided significantly in his pursuit of a Bachelor of Science Degree in Electrical and Computer Engineering from OSU in 1987. His first assignment sent him back home to Chanute AFB for technical training school to become an aircraft maintenance officer. Six months later he was sent to Kirtland AFB, NM to being "pounding the flightline" as a supervisor. He ended this tour with a deployment to Rhein-Main, GE during Operations Desert Shield/Storm where he was the Officer In Charge of the 435th Aircraft Maintenance Unit. From June 1991 to December 1992, he attended the Air Force Institute of Technology (AFIT) where he received a Master of Science Degree in Electrical Engineering with emphasis in navigation and control systems. Following AFIT, he was married to his lovely wife, Angela, and was assigned to Space Division, Los Angeles AFB, CA to develop the future generation of weather satellites. He returned to AFIT in May 1995 to pursue a Doctorate of Philosophy in Electrical Engineering.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE May 1998		3. REPORT TYPE AND DATES COVERED Doctoral Dissertation
4. TITLE AND SUBTITLE New Algorithms for Moving-Bank Multiple Model Adaptive Estimation			5. FUNDING NUMBERS	
6. AUTHOR(S) Juan R. Vasquez, Captain, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB, OH 45433-6583			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/DS/ENG/98-10	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Ms. Sandra Berning AFRL/ SNAR 2241 Avionics Circle, WPAFB OH 45433-7318			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>The focus of this research is to provide methods for generating precise parameter estimates in the face of potentially significant parameter variations such as system component failures. The standard Multiple Model Adaptive Estimation (MMAE) algorithm uses a bank of Kalman filters, each based on a different model of the system. A new moving-bank MMAE algorithm is developed based on exploitation of the density data available from the MMAE. The methods used to exploit this information include various measures of the density data and a decision-making logic used to move, expand, and contract the MMAE bank of filters. Parameter discretization within the MMAE refers to selection of the parameter values assumed by the elemental Kalman filters. A new parameter discretization method is developed based on the probabilities associated with the generalized Chi-Squared random variables formed by residual information from the elemental Kalman filters within the MMAE. Modifications to an existing discretization method are also presented, permitting application of this method in real time and to nonlinear system models or linear/linearized models that are unstable or astable. These new algorithms are validated through computer simulation of an aircraft navigation system subjected to interference/jamming while attempting a successful precision landing of the aircraft.</p>				
14. SUBJECT TERMS Kalman Filtering State Estimation Parameter Estimation Multiple Model Adaptive Estimation			15. NUMBER OF PAGES 324 16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	